E6290

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-085546
(43)Date of publication of application : 30.03.1999

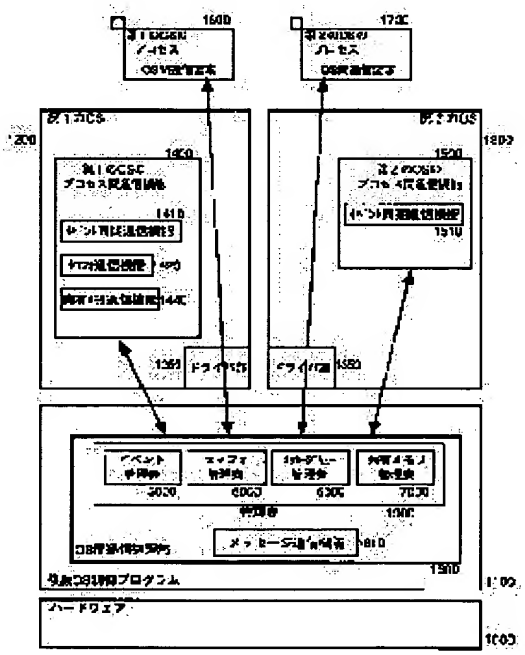
(51)Int.CI. G06F 9/46
G06F 9/46

(21)Application number : 09-248176
(22)Date of filing : 12.09.1997

(71)Applicant : HITACHI LTD
(72)Inventor : INOUE TARO
ARAI TOSHIAKI
KANEKO SHIGENORI
ONO HIROSHI
SEKIGUCHI TOMONORI
SHIBATA TAKASHI

(54) INTER-PROCESS COMMUNICATING METHOD ON HETEROGENEOUS OS

(57)Abstract:
PROBLEM TO BE SOLVED: To perform process synchronization and communication between processes on respective OSs by providing a communication means to communicate each other on 1st and 2nd OSs which simultaneously operate on one computer.
SOLUTION: A management table 1900 which records the name of an inter-OS communication object and the correspondence of an object handle about an inter-process communication function of a 1st OS 1200 and an object handle about an inter-process communication function of a 2nd OS 1300 is provided in each inter-OS communication object. By the table 1900, a communication request that specifies the names of inter-OS communication objects which are issued from processes of the 1st OS and the 2nd OS can be converted into a request to an inter-process communication function of the OS 1200 or the OS 1300. That is, for instance, an inter-OS communication request that specifies a semaphore can be converted into a semaphore communication request for the OS 1200.



LEGAL STATUS

[Date of request for examination] 09.03.2001

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

E6290

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平11－85546

(43)公開日　平成11年(1999)３月30日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/46	３５０	Ｇ０６Ｆ　9/46	３５０
	３４０		３４０Ａ

審査請求　未請求　請求項の数45　ＯＬ　（全 38 頁）

(21)出願番号　特願平9－248176

(22)出願日　平成９年(1997)９月12日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　井上　太郎
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(72)発明者　新井　利明
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(72)発明者　金子　茂則
茨城県日立市大みか町五丁目２番１号　株式会社日立製作所大みか工場内

(74)代理人　弁理士　小川　勝男

最終頁に続く

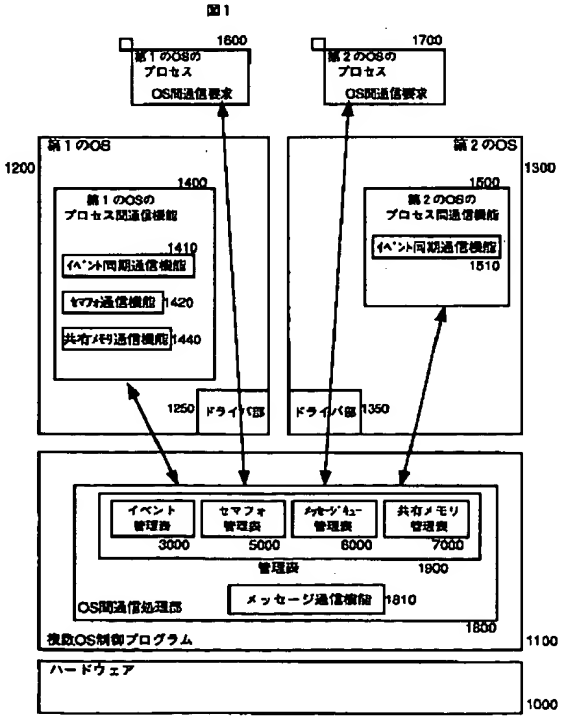
(54)【発明の名称】　異種ＯＳ上プロセス間通信方法

(57)【要約】

【課題】１台の計算機上で複数のオペレーティングシステム（ＯＳ）が動作するとき、それぞれのＯＳ上で動作するプロセス間でのプロセス間通信（イベント同期、セマフォ、メッセージ、共有メモリ）を可能とする。

【解決手段】ＯＳ間通信オブジェクト（イベント、セマフォ、メッセージキュー、共有メモリ等）毎に、ＯＳ間通信オブジェクトの名前と第１のＯＳのプロセス間通信機能に関するオブジェクトハンドルと第２のＯＳのプロセス間通信機能に関するオブジェクトハンドルの対応を記録する管理表を設ける。この管理表により、第１のＯＳ上のプロセスおよび第２のＯＳ上のプロセスから発行されたＯＳ間通信オブジェクトの名前を指定した通信要求を、第１のＯＳあるいは第２のＯＳのプロセス間通信機能への要求へと変換する。

図1



【特許請求の範囲】

【請求項１】１台の計算機上で第１のオペレーティングシステム（ＯＳ）および第２のＯＳが同時に動作する計算機システムにおいて、
第１のＯＳ上のプロセスおよび第２のＯＳ上のプロセスが相互に通信する手段を有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項２】請求項１の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスおよび第２のＯＳ上のプロセスが相互に通信する手段はイベント同期によるプロセス間通信であることを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項３】請求項２の異種ＯＳ上プロセス間通信方法において、
第１のＯＳおよび第２のＯＳが、それぞれ、イベント同期によるプロセス間通信機能を有する時、
ウェイトしている第１のＯＳ上のプロセスを第２のＯＳ上のプロセスがポストし、ウェイトしている第２のＯＳ上のプロセスを第１のＯＳ上のプロセスがポストする手段を有することを特徴とする計算機システム異種ＯＳ上プロセス間通信方法。

【請求項４】請求項３の異種ＯＳ上プロセス間通信方法において、
第１のＯＳのイベント同期によるプロセス間通信機能に関するイベントオブジェクトのハンドルと、第２のＯＳのイベント同期によるプロセス間通信機能に関するイベントオブジェクトのハンドルと、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のイベント同期によるプロセス間通信のためのイベントの名前とのイベント管理表を、第１のＯＳおよび第２のＯＳを制御するための複数ＯＳ制御プログラムの中に有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項５】請求項４の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのオープンの要求を当該イベントの名前を指定して行う場合、
当該イベント管理表をサーチして、当該イベントの名前と同一のイベントの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、
当該イベント管理表をサーチして、当該イベントの名前と同一のイベントの名前が格納されたエントリがなければ、第１のＯＳのイベント同期によるプロセス間通信機能におけるオープン要求を発行し、その結果得られるイベントオブジェクトのハンドルおよび当該イベントの名前を当該イベント管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項６】請求項４の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのウェイトの要求を行う場合、
上記のイベント管理表を用いて第１のＯＳのイベント同期によるプロセス間通信のイベントオブジェクトハンドルを得るステップと、
上記の獲得したイベントオブジェクトハンドルを用いて、第１のＯＳのイベント同期によるプロセス間通信機能におけるウェイトの要求を発行するステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項７】請求項４の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのポストの要求を行う場合、
上記のイベント管理表を用いて第２のＯＳのイベント同期によるプロセス間通信のイベントオブジェクトハンドルを得るステップと、
第１のＯＳのコンテキストから第２のＯＳのコンテキストに切り替えた後に、上記の獲得したイベントオブジェクトハンドルを用いて、第２のＯＳのイベント同期によるプロセス間通信機能におけるポストの要求を発行するステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項８】請求項４の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのクローズの要求を行う場合、
上記のイベント管理表を用いて第１のＯＳのイベント同期によるプロセス間通信のイベントオブジェクトハンドルを得るステップと、
上記の獲得したイベントオブジェクトハンドルを用いて、第１のＯＳのイベント同期によるプロセス間通信機能におけるクローズの要求を発行するステップと、
イベント管理表の当該エントリを無効とするステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項９】請求項４の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのオープンの要求を当該イベントの名前を指定して行う場合、
当該イベント管理表をサーチして、当該イベントの名前

と同一のイベントの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、
当該イベント管理表をサーチして、当該イベントの名前と同一のイベントの名前が格納されたエントリがなければ、第２のＯＳのイベント同期によるプロセス間通信機能におけるオープン要求を発行し、その結果得られるイベントオブジェクトのハンドルおよび当該イベントの名前を当該イベント管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項１０】請求項４の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのウェイトの要求を行う場合、
上記のイベント管理表を用いて第２のＯＳのイベント同期によるプロセス間通信のイベントオブジェクトハンドルを得るステップと、
上記の獲得したイベントオブジェクトハンドルを用いて、第２のＯＳのイベント同期によるプロセス間通信機能におけるウェイトの要求を発行するステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項１１】請求項４の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのポストの要求を行う場合、
上記のイベント管理表を用いて第１のＯＳのイベント同期によるプロセス間通信のイベントオブジェクトハンドルを得るステップと、
第２のＯＳのコンテキストから第１のＯＳのコンテキストに切り替えた後に、上記の獲得したイベントオブジェクトハンドルを用いて、第１のＯＳのイベント同期によるプロセス間通信機能におけるポストの要求を発行するステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項１２】請求項４の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのクローズの要求を行う場合、
上記のイベント管理表を用いて第２のＯＳのイベント同期によるプロセス間通信のイベントオブジェクトハンドルを得るステップと、
上記の獲得したイベントオブジェクトハンドルを用いて、第２のＯＳのイベント同期によるプロセス間通信機能におけるクローズの要求を発行するステップと、
イベント管理表の当該エントリを無効とするステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項１３】請求項１の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスおよび第２のＯＳ上のプロセスが相互に通信する手段はセマフォであることを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項１４】請求項１３の異種ＯＳ上プロセス間通信方法において、
第１のＯＳがセマフォ機能を有し、第２のＯＳがセマフォ機能を有さない時、
第１のＯＳ上のプロセスが発行する、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるセマフォ関連要求を、第１のＯＳが有するセマフォ機能を用いて処理するステップと、
第２のＯＳ上のプロセスが発行する、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるセマフォ関連要求を、第１のＯＳが有するセマフォ機能を用いて処理するステップとを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項１５】請求項１４の異種ＯＳ上プロセス間通信方法において、
第１のＯＳのセマフォ機能に関するセマフォオブジェクトのハンドルと、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるセマフォの名前とのセマフォ管理表を、第１のＯＳおよび第２のＯＳを制御するための複数ＯＳ制御プログラムの中に有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項１６】請求項１５の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるセマフォのオープンの要求を当該セマフォの名前を指定して行う場合、
当該セマフォ管理表をサーチして、当該セマフォの名前と同一のセマフォの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、
当該セマフォ管理表をサーチして、当該セマフォの名前と同一のセマフォの名前が格納されたエントリがなければ、第１のＯＳのセマフォ機能におけるオープン要求を発行し、その結果得られるセマフォオブジェクトのハンドルおよび当該セマフォの名前を当該セマフォ管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項１７】請求項１５の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信における

セマフォのウェイトの要求を行う場合、
上記のセマフォ管理表を用いて第１のＯＳのセマフォによるプロセス間通信のセマフォオブジェクトハンドルを得るステップと、
上記の獲得したセマフォオブジェクトハンドルを用いて、第１のＯＳのセマフォ機能におけるセマフォのウェイトの要求を発行するステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項１８】請求項１５の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるセマフォのカウントアップの要求を行う場合、
上記のセマフォ管理表を用いて第１のＯＳのセマフォによるプロセス間通信のセマフォオブジェクトハンドルを得るステップと、
上記の獲得したセマフォオブジェクトハンドルを用いて、第１のＯＳのセマフォ機能におけるセマフォのカウントアップの要求を発行するステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項１９】請求項１５の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるセマフォのクローズの要求を行う場合、
上記のセマフォ管理表を用いて第１のＯＳのセマフォによるプロセス間通信のセマフォオブジェクトハンドルを得るステップと、
上記の獲得したセマフォオブジェクトハンドルを用いて、第１のＯＳのセマフォ機能におけるクローズの要求を発行するステップと、
セマフォ管理表の当該エントリを無効とするステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項２０】請求項１５の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるセマフォのオープンの要求を当該セマフォの名前を指定して行う場合、
当該セマフォ管理表をサーチして、当該セマフォの名前と同一のセマフォの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、
当該セマフォ管理表をサーチして、当該セマフォの名前と同一のセマフォの名前が格納されたエントリがなければ、第２のＯＳのコンテキストから第１のＯＳのコンテキストに切り替えた後に、第１のＯＳのセマフォ機能におけるオープン要求を発行し、その結果得られるセマフォオブジェクトのハンドルおよび当該セマフォの名前を当該セマフォ管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項２１】請求項１５の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるセマフォのウェイトの要求を行う場合、
上記のセマフォ管理表を用いて第１のＯＳのセマフォによるプロセス間通信のセマフォオブジェクトハンドルを得るステップと、
第２のＯＳのコンテキストから第１のＯＳのコンテキストに切り替え後に、上記の獲得したセマフォオブジェクトハンドルを用いて、第１のＯＳのセマフォ機能におけるセマフォのウェイトの要求を発行するステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項２２】請求項１５の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるセマフォのカウントアップの要求を行う場合、
上記のセマフォ管理表を用いて第１のＯＳのセマフォによるプロセス間通信のセマフォオブジェクトハンドルを得るステップと、
第２のＯＳのコンテキストから第１のＯＳのコンテキストに切り替えた後に、上記の獲得したセマフォオブジェクトハンドルを用いて、第１のＯＳのセマフォ機能におけるセマフォのカウントアップの要求を発行するステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項２３】請求項１５の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるセマフォのクローズの要求を行う場合、
上記のセマフォ管理表を用いて第１のＯＳのセマフォによるプロセス間通信のセマフォオブジェクトハンドルを得るステップと、
第２のＯＳのコンテキストから第１のＯＳのコンテキストに切り替えた後に、上記の獲得したセマフォオブジェクトハンドルを用いて、第１のＯＳのセマフォ機能におけるクローズの要求を発行するステップと、
セマフォ管理表の当該エントリを無効とするステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項２４】請求項１の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスおよび第２のＯＳ上のプロセスが相互に通信する手段はメッセージキューを用いたメッセージであることを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項２５】請求項２４の異種ＯＳ上プロセス間通信方法において、
第１のＯＳがメッセージ機能を有さず、第２のＯＳもメッセージ機能を有さない時、
第１のＯＳ上のプロセスが発行する、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるメッセージ関連要求を処理するステップと、
第２のＯＳ上のプロセスが発行する、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるメッセージ関連要求を処理するステップとを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項２６】請求項２５の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるメッセージキューの名前を管理するメッセージキュー管理表を、第１のＯＳおよび第２のＯＳを制御するための複数ＯＳ制御プログラムの中に有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項２７】請求項２６の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるメッセージキューのオープンの要求を当該メッセージキューの名前を指定して行う場合、
当該メッセージキュー管理表をサーチして、当該メッセージキューの名前と同一のメッセージキューの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、
当該メッセージキュー管理表をサーチして、当該メッセージキューの名前と同一のメッセージキューの名前が格納されたエントリがなければ、新たなメッセージキューを確保して、当該メッセージキューの名前を当該メッセージキュー管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項２８】請求項２６の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるメッセージ送信の要求を行う場合、
第１のＯＳ上のプロセスの空間上のメッセージデータを、メッセージキューが存在する空間上のバッファへコピーするステップと、
メッセージヘッダを割り当てて、それをメッセージキューのメッセージヘッダリストの最後に連結するステップと、
当該メッセージキューに対するメッセージ到着をウェイトしている第２のＯＳ上のプロセスをポストするステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項２９】請求項２６の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるメッセージ受信の要求を行う場合、
当該受信要求において指定したメッセージキューの最初のメッセージを、当該第１のＯＳ上のプロセスの空間の当該受信要求において指定されたアドレスへコピーするステップと、
当該受信要求において指定したメッセージキューにメッセージがなかった場合には、当該メッセージキューに対するメッセージ到着をウェイトするステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項３０】請求項２６の異種ＯＳ上プロセス間通信方法において、
第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるメッセージキューのクローズの要求を行う場合、
メッセージキュー管理表の当該エントリを無効とするステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項３１】請求項２６の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるメッセージキューのオープンの要求を当該メッセージキューの名前を指定して行う場合、
当該メッセージキュー管理表をサーチして、当該メッセージキューの名前と同一のメッセージキューの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、
当該メッセージキュー管理表をサーチして、当該メッセージキューの名前と同一のメッセージキューの名前が格納されたエントリがなければ、新たなメッセージキューを確保して、当該メッセージキューの名前を当該メッセージキュー管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項３２】請求項２６の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるメッセージ送信の要求を行う場合、
第２のＯＳ上のプロセスの空間上のメッセージデータを、メッセージキューが存在する空間上のバッファへコピーするステップと、
メッセージヘッダを割り当てて、それをメッセージキューのメッセージヘッダリストの最後に連結するステップと、

当該メッセージキューに対するメッセージ到着をウェイトしている第１のＯＳ上のプロセスをポストするステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項３３】請求項２６の異種ＯＳ上プロセス間通信方法において、

第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるメッセージ受信の要求を行う場合、

当該受信要求において指定したメッセージキューの最初のメッセージを、当該第２のＯＳ上のプロセスの空間の当該受信要求において指定されたアドレスへコピーするステップと、

当該受信要求において指定したメッセージキューにメッセージがなかった場合には、当該メッセージキューに対するメッセージ到着をウェイトするステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項３４】請求項２６の異種ＯＳ上プロセス間通信方法において、

第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信におけるメッセージキューのクローズの要求を行う場合、

メッセージキュー管理表の当該エントリを無効とするステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項３５】請求項１の異種ＯＳ上プロセス間通信方法において、

第１のＯＳ上のプロセスおよび第２のＯＳ上のプロセスが相互に通信する手段は共有メモリであることを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項３６】請求項３５の異種ＯＳ上プロセス間通信方法において、

第１のＯＳが共有メモリ機能を有し、第２のＯＳが共有メモリ機能を有さない時、

第１のＯＳ上のプロセスが発行する、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信における共有メモリ関連要求を、第１のＯＳが有する共有メモリ機能を用いて処理するステップと、

第２のＯＳ上のプロセスが発行する、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信における共有メモリ関連要求を、第１のＯＳが有する共有メモリ機能を用いて処理するステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項３７】請求項３６の異種ＯＳ上プロセス間通信方法において、

第１のＯＳの共有メモリに関する共有メモリオブジェクトのハンドルと、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信における共有メモリの名前との共有メモリ管理表を、第１のＯＳおよび第２のＯＳを制御するための複数ＯＳ制御プログラムの中に有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項３８】請求項３７の異種ＯＳ上プロセス間通信方法において、

第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信における共有メモリのオープンの要求を当該共有メモリの名前を指定して行う場合、

当該共有メモリ管理表をサーチして、当該共有メモリの名前と同一の共有メモリの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、

当該共有メモリ管理表をサーチして、当該共有メモリの名前と同一の共有メモリの名前が格納されたエントリがなければ、第１のＯＳの共有メモリ機能におけるオープン要求を発行し、その結果得られる共有メモリオブジェクトのハンドルおよび当該共有メモリの名前を当該共有メモリ管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項３９】請求項３７の異種ＯＳ上プロセス間通信方法において、

第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信における共有メモリの仮想空間マッピングの要求を行う場合、

上記のセマフォ管理表を用いて第１のＯＳの共有メモリによるプロセス間通信の共有メモリオブジェクトハンドルを得るステップと、

上記の獲得した共有メモリオブジェクトハンドルを用いて、第１のＯＳの共有メモリ機能における共有メモリの仮想空間マッピングの要求を発行するステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項４０】請求項３７の異種ＯＳ上プロセス間通信方法において、

第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信における共有メモリの仮想空間マッピング解除の要求を行う場合、

上記の共有メモリ管理表を用いて第１のＯＳの共有メモリによるプロセス間通信の共有メモリオブジェクトハンドルを得るステップと、

上記の獲得した共有メモリオブジェクトハンドルを用いて、第１のＯＳの共有メモリ機能における共有メモリの仮想空間マッピング解除の要求を発行するステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。

【請求項４１】請求項３７の異種ＯＳ上プロセス間通信方法において、

第１のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信における共有メモリのクローズの要求を行う場合、

上記の共有メモリ管理表を用いて第１のＯＳの共有メモ

リによるプロセス間通信の共有メモリオブジェクトハンドルを得るステップと、
上記の獲得した共有メモリオブジェクトハンドルを用いて、第１のＯＳの共有メモリ機能におけるクローズの要求を発行するステップと、
共有メモリ管理表の当該エントリを無効とするステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項４２】請求項３７の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信における共有メモリのオープンの要求を当該共有メモリの名前を指定して行う場合、
当該共有メモリ管理表をサーチして、当該共有メモリの名前と同一の共有メモリの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、
当該共有メモリ管理表をサーチして、当該共有メモリの名前と同一の共有メモリの名前が格納されたエントリがなければ、第２のＯＳのコンテキストから第１のＯＳのコンテキストに切り替えた後に、第１のＯＳの共有メモリ機能におけるオープン要求を発行し、その結果得られる共有メモリオブジェクトのハンドルおよび当該共有メモリの名前を当該共有メモリ管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項４３】請求項３７の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信における共有メモリの仮想空間マッピングの要求を行う場合、
上記の共有メモリ管理表を用いて第１のＯＳの共有メモリによるプロセス間通信の共有メモリオブジェクトハンドルを得るステップと、
第２のＯＳのコンテキストから第１のＯＳのコンテキストに切り替え後に、上記の獲得した共有メモリオブジェクトハンドルを用いて、第１のＯＳの共有メモリ機能における共有メモリの仮想空間マッピングの要求を発行するステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項４４】請求項３７の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信における共有メモリの仮想空間マッピング解除の要求を行う場合、
上記の共有メモリ管理表を用いて第１のＯＳの共有メモリによるプロセス間通信の共有メモリオブジェクトハンドルを得るステップと、
第２のＯＳのコンテキストから第１のＯＳのコンテキストに切り替えた後に、上記の獲得した共有メモリオブジェクトハンドルを用いて、第１のＯＳの共有メモリ機能における共有メモリの仮想空間マッピング解除の要求を発行するステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【請求項４５】請求項３７の異種ＯＳ上プロセス間通信方法において、
第２のＯＳ上のプロセスが、第１のＯＳ上のプロセスと第２のＯＳ上のプロセスの間のプロセス間通信における共有メモリのクローズの要求を行う場合、
上記の共有メモリ管理表を用いて第１のＯＳの共有メモリによるプロセス間通信の共有メモリオブジェクトハンドルを得るステップと、
第２のＯＳのコンテキストから第１のＯＳのコンテキストに切り替えた後に、上記の獲得した共有メモリオブジェクトハンドルを用いて、第１のＯＳの共有メモリ機能におけるクローズの要求を発行するステップと、
共有メモリ管理表の当該エントリを無効とするステップを有することを特徴とする異種ＯＳ上プロセス間通信方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、１台の計算機上で同時に動作する異なるオペレーティングシステム上のプロセスが相互に同期および通信する方法に関する。
【０００２】
【従来の技術】一般的に、計算機では１つのオペレーティングシステム（ＯＳ）が動作し、それが計算機の各種資源（ＣＰＵ、メモリ、２次記憶装置等）を管理している。このＯＳには様々な種類のものがあり、それぞれに特徴がある。即ち、オンライン処理に向いたＯＳ、リアルタイム処理向けのＯＳ、グラフィカルユーザインタフェース（ＧＵＩ）に優れたＯＳ等である。すると１台の計算機上に複数のＯＳを動作させようとする要求が生まれる。このための技術としては、仮想計算機あるいはマイクロカーネルがある。
【０００３】仮想計算機は、主に汎用計算機の世界で実現されている技術である。仮想計算機制御プログラムがすべての計算機資源を管理し、それを各仮想計算機に対して分割専有したり、仮想化して与えたりするのである。
【０００４】一方、マイクロカーネルでは、マイクロカーネルの上にＯＳ機能を実現するＯＳサーバを構築し、ユーザはそのサーバを通して計算機を利用する。このＯＳサーバを各種用意すれば様々なＯＳ環境を利用できる。
【０００５】また、計算機のＯＳが提供する機能として、プロセス間の同期機能および通信機能がある（以下では、「プロセス間の同期機能および通信機能」のことを「プロセス間同期・通信機能」と呼ぶことがある）。

プロセス間の同期機能は、主に複数のプロセスが資源を共有する場合にシリアライズして競合を防止するための機能であり、プロセス間通信機能は、複数のプロセスが互いにデータのやりとりをするための機能である。例えば、プロセス間の同期機能としてはイベント同期機能やセマフォがあり、プロセス間通信機能としてはメッセージや共有メモリがある（「ＵＮＩＸカーネルの設計」Maurice J. Bach 著、坂本文・多田好克・村井純 訳、ｐ３０４～３２２、原題 "The Design of the UNIX Operating System"）。

【０００６】

【発明が解決しようとする課題】以上のような仮想計算機あるいはマイクロカーネル等の技術により１台の計算機上に複数のＯＳを動作させることが可能となる。すると、例えば、１台の計算機上でリアルタイム処理向けＯＳとＧＵＩに優れたＯＳが動作しているとき、リアルタイム処理向けＯＳ上のプロセスが取得したデータをＧＵＩに優れたＯＳ上のプロセスで処理して画面に表示する、といった要求が生まれる。これを実現するためには、同時に動作する複数のＯＳ上のプロセス同士の同期機能および通信機能が必要となる。

【０００７】そこで、本発明の第１の目的は、１台の計算機上で複数のＯＳが動作する場合に、それぞれのＯＳ上のプロセス間でのプロセス間同期・通信機能を実現するための方法を提供することにある。

【０００８】また、異種ＯＳ上のプロセス間での同期機能あるいは通信機能を実現する場合、このＯＳが備えているプロセス間同期・通信機能を利用することができれば、実現が容易となる。

【０００９】そこで、本発明の第２の目的は、１台の計算機上で複数のＯＳが動作し、それぞれのＯＳ上のプロセス相互間のプロセス間同期・通信機能を実現する際に、これらのＯＳがもともと備えているプロセス間同期・通信機能を用いて実現するための方法を提供することにある。

【００１０】

【課題を解決するための手段】本発明では、ＯＳ間通信オブジェクト（イベント、セマフォ、メッセージキュー、共有メモリ等）毎に、ＯＳ間通信オブジェクトの名前と第１のＯＳのプロセス間通信機能に関するオブジェクトハンドルと第２のＯＳのプロセス間通信機能に関するオブジェクトハンドルの対応を記録する管理表を設ける。この管理表により、第１のＯＳ上のプロセスおよび第２のＯＳ上のプロセスから発行されたＯＳ間通信オブジェクトの名前を指定した通信要求を、第１のＯＳあるいは第２のＯＳのプロセス間通信機能への要求へと変換できる。即ち、例えば、セマフォを指定したＯＳ間通信要求を第１のＯＳのセマフォ通信要求に変換できるのである。これにより、例えば、イベント同期によるプロセス間通信の場合では、ウェイトしている第１のＯＳ上のプロセスを第２のＯＳ上のプロセスがポストし、ウェイトしている第２のＯＳ上のプロセスを第１のＯＳ上のプロセスがポストすることが可能となる。

【００１１】

【発明の実施の形態】以下、本発明の実施例を、図面により詳細に説明する。

【００１２】図１は、本発明の実施例を実施する計算機システムにおけるＯＳ間通信の概要を示す図である。ハードウェア１０００は、この計算機システムを構成するハードウェアであり、ＣＰＵ、メモリ、入出力装置等から成る。複数ＯＳ制御プログラム１１００は、この計算機上で複数のＯＳ（この例では第１のＯＳ１２００および第２のＯＳ１３００）が動作するための各種の制御をする部分で、ハードウェア１０００の初期化および分割専有処理、ＣＰＵのスケジューリング、割り込み処理等を行う。ＯＳ間通信処理部１８００は、複数ＯＳ制御プログラム１１００の中に存在し、管理表１９００を用いてこの計算機システム上で動作するＯＳのＯＳ間通信の処理を行う。管理表１９００は、ＯＳ間通信オブジェクト対応に存在し、イベント管理表３０００、セマフォ管理表５０００、メッセージキュー管理表６０００、共有メモリ管理表７０００がある。そして、ＯＳ間通信処理部１８００には、メッセージ通信機能１８１０がある。このメッセージ通信機能１８１０は、メッセージによるＯＳ間通信の処理を行う。第１のＯＳ１２００および第２のＯＳ１３００は、この計算機システム上で動作するオペレーティングシステム（ＯＳ）で、例えば、第１のＯＳがグラフィカルユーザインタフェース（ＧＵＩ）に優れたＯＳであり、第２のＯＳがリアルタイム性に優れたＯＳであるという具合である。これらのＯＳは各々ドライバ部１２５０およびドライバ部１３５０を含んでおり、これらドライバを経由してＯＳ間通信処理部１８００とアクセスする。そして、それぞれのＯＳには、第１のＯＳのプロセス間通信機能１４００および第２のＯＳのプロセス間通信機能１５００がある。これらは、それぞれのＯＳ自身がその上で動作するプロセスに対するプロセス間通信の処理を行う部分であり、第１のＯＳのプロセス間通信機能１４００は、イベント同期通信機能１４１０、セマフォ通信機能１４２０、共有メモリ通信機能１４４０から成る。第２のＯＳのプロセス間通信機能１５００は、イベント同期通信機能１５１０から成る。第１のＯＳのプロセス１６００は第１のＯＳ１２００の上で動作するプロセスであり、第２のＯＳのプロセス１７００は第２のＯＳ１３００の上で動作するプロセスである。第１のＯＳのプロセス１６００あるいは第２のＯＳのプロセス１７００からＯＳ間通信オブジェクトのオープンの要求がドライバ部１２５０あるいはドライバ部１３５０を経由して出されると、ＯＳ間通信処理部１８００は、この要求で指定されたＯＳ間通信オブジェクトの名前を管理表１９００から探し、これと同じ名前のＯ

Ｓ間通信オブジェクトのエントリがあればその記述子を返し、同じ名前のＯＳ間通信オブジェクトがなければ、第１のＯＳのプロセス間通信機能１４００に対してオープン要求を行う。そして、返されたオブジェクトハンドルとこのＯＳ間通信オブジェクトの名前を管理表１９００に登録し、そのエントリの記述子を返す。

【００１３】図２には、第１のＯＳのプロセス１６００および第２のＯＳのプロセス１７００のプロセスのアドレス空間の構成を示す。第１のＯＳのプロセス１６００のアドレス空間２１００および第２のＯＳのプロセス１７００のプロセスのアドレス空間２２００は、下半分が各アプリケーション領域２３００である。そして、上半分はシステム領域２４００であり、第１のＯＳのプロセス１６００および第２のＯＳのプロセス１７００の両者から共通な領域である。このシステム領域２４００には、複数ＯＳ制御部１１００およびＯＳ間通信処理部１８００が存在する。

【００１４】以下では、まず、イベント同期に関連した処理について説明する。

【００１５】図３は、管理表１９００の中のイベント管理表３０００を示す図である。イベント管理表３０００は、ＯＳ間通信処理部１８００の初期処理時に確保される。イベント管理表３０００は、エントリ有効フラグ３０１０、イベントの名前３０２０、第１のＯＳのイベントオブジェクトハンドル３０３０、第１のＯＳのイベントオブジェクトハンドル有効フラグ３０４０、第２のＯＳのイベントオブジェクトハンドル３０５０、第２のＯＳのイベントオブジェクトハンドル有効フラグ３０６０から成る。この場合は、第１のＯＳ１２００および第２のＯＳ１３００がともにイベント同期通信機能を有するので、第１のＯＳ１２００のイベント同期通信機能のオブジェクトハンドルを格納するのがイベントオブジェクトハンドル３０３０であり、イベントオブジェクトハンドル有効フラグ３０４０はそれの有効か無効かを示すフラグである。そして、第２のＯＳ１３００のイベント同期通信機能のオブジェクトハンドルを格納するのが第２のＯＳのイベントオブジェクトハンドル３０５０で、第２のＯＳのイベントオブジェクトハンドル有効フラグ３０６０はそれの有効か無効かを示すフラグである。また、エントリ有効フラグ３０１０は、このエントリが有効か無効かを示すフラグである。イベントの名前３０２０は、ＯＳ間通信オブジェクトであるところのイベントの名前を格納する。

【００１６】ＯＳの中には、プロセス間のイベント同期のためにＥＣＢ（イベントコントロールブロック）を使用するものもあるが、そのような場合には、第１のＯＳのイベントオブジェクトハンドル３０３０あるいは第２のＯＳのイベントオブジェクトハンドル３０５０に、ＥＣＢのアドレスを格納する。

【００１７】図４は、第１のＯＳのプロセス１６００が発行したイベント同期通信のオープン要求システムコールの処理のフローである。このシステムコールは、イベントの名前をパラメータに指定してオープン要求を行い、その結果としてイベント記述子を得る。以降のイベント同期通信関連のシステムコールでは、このイベント記述子を用いることになる。まず、イベント管理表３０００を調べて、エントリ有効フラグ３０１０がＯＮで、かつ、指定されたイベントの名前とイベントの名前３０２０が同じであるエントリを探す（ステップ４０１０）。イベント管理表３０００にこのようなエントリがあれば、既にこの名前のイベントはオープンされていることになる。そこで、このエントリについて、第１のＯＳのイベントオブジェクトハンドル有効フラグ３０４０を調べ（ステップ４０９０）、ＯＮなら、そのエントリのイベント記述子を返して終了する（ステップ４１００）。ＯＦＦなら、第１のＯＳ１２００のイベント同期通信機能１４１０のオープンのシステムコールをこのイベントの名前で発行することにより、第１のＯＳ１２００のイベント同期通信機能１４１０におけるオブジェクトハンドルを獲得し（ステップ４１１０）、獲得したオブジェクトハンドルを第１のＯＳのイベントオブジェクトハンドル３０３０にセットし（ステップ４１２０）、第１のＯＳのイベントオブジェクトハンドル有効フラグ３０４０をＯＮにし（ステップ４１３０）、そのエントリのイベント記述子を返して終了する（ステップ４１４０）。

【００１８】一方、ステップ４０１０で、エントリ有効フラグ３０１０がＯＮで、かつ、指定されたイベントの名前とイベントの名前３０２０が同じであるエントリが、イベント管理表３０００になければ、エントリ有効フラグ３０１０がＯＦＦのエントリを探す（ステップ４０２０）。なければ、イベント管理表３０００のすべてのエントリが使用中であることになるのでエラーリターンする（ステップ４０３０）。エントリ有効フラグ３０１０がＯＦＦのエントリがあれば、第１のＯＳ１２００のイベント同期通信１４１０のオープンのシステムコールをこのイベントの名前で発行することにより、第１のＯＳ１２００のイベント同期通信機能１４１０のオブジェクトハンドルを獲得し（ステップ４０４０）、獲得したオブジェクトハンドルを第１のＯＳのイベントオブジェクトハンドル３０３０にセットし（ステップ４０５０）、第１のＯＳのイベントオブジェクトハンドル有効フラグ３０４０をＯＮにし（ステップ４０６０）、パラメータで指定されたイベントの名前をイベントの名前３０２０に格納し（ステップ４０６５）、エントリ有効フラグ３０１０をＯＮにして（ステップ４０７０）、そのエントリのイベント記述子を返して終了する（ステップ４０８０）。

【００１９】図５は、第１のＯＳのプロセス１６００が発行したイベント同期通信のウェイト要求システムコー

ルの処理のフローである。このシステムコールは、イベント記述子をパラメータに指定してウェイト要求を行う。まず、当該ウェイト要求において指定されたイベント記述子に対応するエントリをイベント管理表３０００からみつけ、そのエントリにおける第１のＯＳのイベントオブジェクトハンドル３０３０を用いて、第１のＯＳ１２００のイベント同期通信機能１４１０のウェイト要求のシステムコールを発行し（ステップ４２１０）、リターンする（ステップ４２２０）。

【００２０】図６は、第１のＯＳのプロセス１６００が発行したイベント同期通信のポスト要求システムコールの処理のフローである。このシステムコールは、イベント記述子をパラメータに指定してポスト要求を行う。まず、第２のＯＳ１３００へのコンテキスト切替えを行い（ステップ４３１０）、当該ポスト要求において指定されたイベント記述子に対応するエントリをイベント管理表３０００からみつけ、そのエントリにおける第２のＯＳのイベントオブジェクトハンドル３０５０を用いて、第２のＯＳ１３００のイベント同期通信機能１５１０のポスト要求のシステムコールを発行する（ステップ４３２０）。そして、第１のＯＳ１２００へのコンテキスト切替えを行い（ステップ４３３０）、リターンする（ステップ４３４０）。

【００２１】図７は、第１のＯＳのプロセス１６００が発行したイベント同期通信のクローズ要求システムコールの処理のフローである。このシステムコールは、イベント記述子をパラメータに指定してクローズ要求を行う。まず、当該クローズ要求において指定されたイベント記述子に対応するエントリをイベント管理表３０００からみつけ、そのエントリにおける第１のＯＳのイベントオブジェクトハンドル３０３０を用いて、第１のＯＳ１２００のイベント同期通信機能１４１０のクローズ要求のシステムコールを発行し（ステップ４４１０）、第１のＯＳのイベントオブジェクトハンドル有効フラグ３０４０をＯＦＦにする（ステップ４４２０）。第２のＯＳのイベントオブジェクトハンドル有効フラグ３０６０がＯＦＦなら、エントリ有効フラグ３０１０をＯＦＦにして当該エントリを無効にして（ステップ４４４０）、リターンする（ステップ４４５０）。一方、第２のＯＳのイベントオブジェクトハンドル有効フラグ３０６０がＯＮなら、リターンする（ステップ４４５０）。

【００２２】図８は、第２のＯＳのプロセス１７００が発行したイベント同期通信のオープン要求システムコールの処理のフローである。このシステムコールは、イベントの名前をパラメータに指定してオープン要求を行い、その結果としてイベント記述子を得る。以降のイベント同期通信関連のシステムコールでは、このイベント記述子を用いることになる。まず、イベント管理表３０００を調べて、エントリ有効フラグ３０１０がＯＮで、かつ、指定されたイベントの名前とイベントの名前３０２０が同じであるエントリを探す（ステップ４５１０）。イベント管理表３０００にこのようなエントリがあれば、既にこの名前のイベントはオープンされていることになる。そこで、このエントリについて、第２のＯＳのイベントオブジェクトハンドル有効フラグ３０６０を調べ（ステップ４５９０）、ＯＮなら、そのエントリのイベント記述子を返して終了する（ステップ４６００）。ＯＦＦなら、第２のＯＳ１３００のイベント同期通信１５１０のオープンのシステムコールをこのイベントの名前で発行することにより、第２のＯＳ１３００のイベント同期通信機能１５１０のオブジェクトハンドルを獲得し（ステップ４６１０）、獲得したオブジェクトハンドルを第２のＯＳのイベントオブジェクトハンドル３０５０にセットし（ステップ４６２０）、第２のＯＳのイベントオブジェクトハンドル有効フラグ３０６０をＯＮにし（ステップ４６３０）、そのエントリのイベント記述子を返して終了する（ステップ４６４０）。

【００２３】一方、ステップ４０１０で、エントリ有効フラグ３０１０がＯＮで、かつ、指定されたイベントの名前とイベントの名前３０２０が同じであるエントリが、イベント管理表３０００になければ、エントリ有効フラグ３０１０がＯＦＦのエントリを探す（ステップ４５２０）。なければ、イベント管理表３０００のすべてのエントリが使用中であることになるのでエラーリターンする（ステップ４５３０）。エントリ有効フラグ３０１０がＯＦＦのエントリがあれば、第１のＯＳ１２００のイベント同期通信機能１５１０のオープンのシステムコールをこのイベントの名前で発行することにより、第２のＯＳ１３００のイベント同期通信機能１５１０のオブジェクトハンドルを獲得し（ステップ４５４０）、獲得したオブジェクトハンドルを第２のＯＳのイベントオブジェクトハンドル３０５０にセットし（ステップ４５５０）、第２のＯＳのイベントオブジェクトハンドル有効フラグ３０６０をＯＮにし（ステップ４５６０）、パラメータで指定されたイベントの名前をイベントの名前３０２０に格納し（ステップ４５６５）、エントリ有効フラグ３０１０をＯＮにして（ステップ４５７０）、そのエントリのイベント記述子を返して終了する（ステップ４５８０）。

【００２４】図９は、第２のＯＳのプロセス１７００が発行したイベント同期通信のウェイト要求システムコールの処理のフローである。このシステムコールは、イベント記述子をパラメータに指定してウェイト要求を行う。まず、当該ウェイト要求において指定されたイベント記述子に対応するエントリをイベント管理表３０００からみつけ、そのエントリにおける第２のＯＳのイベントオブジェクトハンドル３０５０を用いて、第２のＯＳ１３００のイベント同期通信機能１５１０のウェイト要求のシステムコールを発行し（ステップ４７１０）、リターンする（ステップ４７２０）。

【００２５】図１０は、第２のＯＳのプロセス１７００が発行したイベント同期通信のポスト要求システムコールの処理のフローである。このシステムコールは、イベント記述子をパラメータに指定してポスト要求を行う。まず、第１のＯＳ１２００へのコンテキスト切替えを行い（ステップ４８１０）、当該ポスト要求において指定されたイベント記述子に対応するエントリをイベント管理表３０００からみつけ、そのエントリにおける第１のＯＳのイベントオブジェクトハンドル３０３０を用いて、第１のＯＳ１２００のイベント同期通信機能１４１０のポスト要求のシステムコールを発行する（ステップ４８２０）。そして、第２のＯＳ１３００へのコンテキスト切替えを行い（ステップ４８３０）、リターンする（ステップ４８４０）。

【００２６】図１１は、第２のＯＳのプロセス１７００が発行したイベント同期通信のクローズ要求システムコールの処理のフローである。このシステムコールは、イベント記述子をパラメータに指定してクローズ要求を行う。まず、当該クローズ要求において指定されたイベント記述子に対応するエントリをイベント管理表３０００からみつけ、そのエントリにおける第２のＯＳのイベントオブジェクトハンドル３０５０を用いて、第２のＯＳ１３００のイベント同期通信機能１５１０のクローズ要求のシステムコールを発行し（ステップ４９１０）、第２のＯＳのイベントオブジェクトハンドル有効フラグ３０６０をＯＦＦにする（ステップ４９２０）。第１のＯＳのイベントオブジェクトハンドル有効フラグ３０４０がＯＦＦなら、エントリ有効フラグ３０１０をＯＦＦにして当該エントリを無効にして（ステップ４９４０）、リターンする（ステップ４９５０）。一方、第２のＯＳのイベントオブジェクトハンドル有効フラグ３０４０がＯＮなら、リターンする（ステップ４９５０）。

【００２７】以上の図３から図１１に示した処理により、第１のＯＳ１２００および第２のＯＳ１３００が、それぞれ、イベント同期によるプロセス間通信機能を有する時、ウェイトしている第１のＯＳ上のプロセス１６００を第２のＯＳ上のプロセス１７００がポストし、ウェイトしている第２のＯＳ上のプロセス１７００を第１のＯＳ上のプロセス１６００がポストすることが可能となる。

【００２８】次に、セマフォに関連した処理について説明する。

【００２９】図１２は、管理表１９００の中のセマフォ管理表５０００を示す図である。セマフォ管理表５０００は、ＯＳ間通信処理部１８００の初期処理時に確保される。セマフォ管理表５０００は、エントリ有効フラグ５０１０、セマフォの名前５０２０、第１のＯＳのセマフォオブジェクトハンドル５０３０、第１のＯＳのセマフォオブジェクトハンドル有効フラグ５０４０から成る。この場合は、第１のＯＳ１２００がセマフォ通信機能を有するので、第１のＯＳ１２００のセマフォ通信機能のオブジェクトハンドルを格納するのがセマフォオブジェクトハンドル５０３０であり、セマフォオブジェクトハンドル有効フラグ５０４０はそれの有効か無効かを示すフラグである。また、エントリ有効フラグ５０１０は、このエントリが有効か無効かを示すフラグである。セマフォの名前５０２０は、ＯＳ間通信オブジェクトであるところのセマフォの名前を格納する。

【００３０】図１３は、第１のＯＳのプロセス１６００が発行したセマフォ通信のオープン要求システムコールの処理のフローである。このシステムコールは、セマフォの名前をパラメータに指定してオープン要求を行い、その結果としてセマフォ記述子を得る。以降のセマフォ通信関連のシステムコールでは、このセマフォ記述子を用いることになる。第１のＯＳ１２００はセマフォ通信機能を有する。まず、セマフォ管理表５０００を調べて、エントリ有効フラグ５０１０がＯＮで、かつ、指定されたセマフォの名前とセマフォの名前５０２０が同じであるエントリを探す（ステップ５１１０）。セマフォ管理表５０００にこのようなエントリがあれば、既にこの名前のセマフォはオープンされていることになる。そこで、そのエントリのセマフォ記述子を返して終了する（ステップ５１９０）。

【００３１】一方、ステップ５１１０で、エントリ有効フラグ５０１０がＯＮで、かつ、指定されたセマフォの名前とセマフォの名前５０２０が同じであるエントリが、セマフォ管理表５０００になければ、エントリ有効フラグ５０１０がＯＦＦのエントリを探す（ステップ５１２０）。なければ、セマフォ管理表５０００のすべてのエントリが使用中であることになるのでエラーリターンする（ステップ５１３０）。エントリ有効フラグ５０１０がＯＦＦのエントリがあれば、第１のＯＳ１２００のセマフォ通信のオープンのシステムコールをこのセマフォの名前で発行することにより、第１のＯＳ１２００のセマフォ通信機能１４２０のオブジェクトハンドルを獲得し（ステップ５１４０）、獲得したオブジェクトハンドルを第１のＯＳのセマフォオブジェクトハンドル５０３０にセットし（ステップ５１５０）、第１のＯＳのセマフォオブジェクトハンドル有効フラグ５０４０をＯＮにし（ステップ５１６０）、パラメータで指定されたセマフォの名前をセマフォの名前５０２０に格納し（ステップ５１６５）、エントリ有効フラグ５０１０をＯＮにして（ステップ５１７０）、そのエントリのセマフォ記述子を返して終了する（ステップ５１８０）。

【００３２】図１４は、第１のＯＳのプロセス１６００が発行したセマフォ通信のセマフォウェイト要求システムコールの処理のフローである。このシステムコールは、セマフォ記述子をパラメータに指定してセマフォウェイト要求を行う。まず、当該セマフォウェイト要求において指定されたセマフォ記述子に対応するエントリを

セマフォ管理表５０００からみつけ、そのエントリにおける第１のＯＳのセマフォオブジェクトハンドル５０３０を用いて、第１のＯＳ１２００のセマフォ通信機能１４２０のセマフォウェイト要求のシステムコールを発行し（ステップ５２１０）、リターンする（ステップ５２２０）。

【００３３】図１５は、第１のＯＳのプロセス１６００が発行したセマフォ通信のセマフォカウントアップ要求システムコールの処理のフローである。このシステムコールは、セマフォ記述子をパラメータに指定してセマフォカウントアップ要求を行う。まず、当該セマフォカウントアップ要求において指定されたセマフォ記述子に対応するエントリをセマフォ管理表５０００からみつけ、そのエントリにおける第１のＯＳのセマフォオブジェクトハンドル５０３０を用いて、第１のＯＳ１２００のセマフォ通信機能１４２０のセマフォカウントアップ要求のシステムコールを発行し（ステップ５３１０）、リターンする（ステップ５３２０）。

【００３４】図１６は、第１のＯＳのプロセス１６００が発行したセマフォ通信のクローズ要求システムコールの処理のフローである。このシステムコールは、セマフォ記述子をパラメータに指定してクローズ要求を行う。まず、当該クローズ要求において指定されたセマフォ記述子に対応するエントリをセマフォ管理表５０００からみつけ、そのエントリにおける第１のＯＳのセマフォオブジェクトハンドル５０３０を用いて、第１のＯＳ１２００のセマフォ通信機能１４２０のクローズ要求のシステムコールを発行し（ステップ５４１０）、第１のＯＳのセマフォオブジェクトハンドル有効フラグ５０４０をＯＦＦにし（ステップ５４２０）、エントリ有効フラグ５０１０をＯＦＦにして当該エントリを無効にして（ステップ５４３０）、リターンする（ステップ５４４０）。

【００３５】図１７は、第２のＯＳのプロセス１７００が発行したセマフォ通信のオープン要求システムコールの処理のフローである。このシステムコールは、セマフォの名前をパラメータに指定してオープン要求を行い、その結果としてセマフォ記述子を得る。以降のセマフォ通信関連のシステムコールでは、このセマフォ記述子を用いることになる。第２のＯＳ１３００はセマフォ通信機能を有しない。まず、セマフォ管理表５０００を調べて、エントリ有効フラグ５０１０がＯＮで、かつ、指定されたセマフォの名前とセマフォの名前５０２０が同じであるエントリを探す（ステップ５５１０）。セマフォ管理表５０００にこのようなエントリがあれば、既にこの名前のセマフォはオープンされていることになる。そこで、そのエントリのセマフォ記述子を返して終了する（ステップ５５９０）。

【００３６】一方、ステップ５５１０で、エントリ有効フラグ５０１０がＯＮで、かつ、指定されたセマフォの名前とセマフォの名前５０２０が同じであるエントリが、セマフォ管理表５０００になければ、エントリ有効フラグ５０１０がＯＦＦのエントリを探す（ステップ５５２０）。なければ、セマフォ管理表５０００のすべてのエントリが使用中であることになるのでエラーリターンする（ステップ５５３０）。エントリ有効フラグ５０１０がＯＦＦのエントリがあれば、第１のＯＳ１２００へコンテキスト切替えを行い（ステップ５５３５）、第１のＯＳ１２００のセマフォ通信機能１４２０のオープンのシステムコールをこのセマフォの名前で発行することにより、第１のＯＳ１２００のセマフォ通信機能１４２０のオブジェクトハンドルを獲得し（ステップ５５４０）、獲得したオブジェクトハンドルを第１のＯＳのセマフォオブジェクトハンドル５０３０にセットし（ステップ５５５０）、第１のＯＳのセマフォオブジェクトハンドル有効フラグ５０４０をＯＮにし（ステップ５５６０）、パラメータとして指定されたセマフォの名前をセマフォの名前５０２０に格納し（ステップ５５６５）、エントリ有効フラグ５０１０をＯＮにして（ステップ５５７０）、第２のＯＳ１３００へコンテキスト切替えを行い（ステップ５５７５）、そのエントリのセマフォ記述子を返して終了する（ステップ５５８０）。

【００３７】図１８は、第２のＯＳのプロセス１７００が発行したセマフォ通信のセマフォウェイト要求システムコールの処理のフローである。このシステムコールは、セマフォ記述子をパラメータに指定してセマフォウェイト要求を行う。まず、第１のＯＳ１２００へのコンテキスト切替えを行い（ステップ５６１０）、当該セマフォウェイト要求において指定されたセマフォ記述子に対応するエントリをセマフォ管理表５０００からみつけ、そのエントリにおける第１のＯＳのセマフォオブジェクトハンドル５０３０を用いて、第１のＯＳ１２００のセマフォ通信機能１４２０のセマフォウェイト要求のシステムコールを発行し（ステップ５６２０）、第２のＯＳ１３００へのコンテキスト切替えを行い（ステップ５６３０）、リターンする（ステップ５６４０）。

【００３８】図１９は、第２のＯＳのプロセス１７００が発行したセマフォ通信のセマフォカウントアップ要求システムコールの処理のフローである。このシステムコールは、セマフォ記述子をパラメータに指定してセマフォカウントアップ要求を行う。まず、第１のＯＳ１２００へのコンテキスト切替えを行い（ステップ５７１０）、当該セマフォカウントアップ要求において指定されたセマフォ記述子に対応するエントリをセマフォ管理表５０００からみつけ、そのエントリにおける第１のＯＳのセマフォオブジェクトハンドル５０３０を用いて、第１のＯＳ１２００のセマフォ通信機能１４２０のセマフォカウントアップ要求のシステムコールを発行し（ステップ５７２０）、第２のＯＳ１３００へのコンテキスト切替えを行い（ステップ５７３０）、リターンする

（ステップ５７４０）。

【００３９】図２０は、第２のＯＳのプロセス１７００が発行したセマフォ通信のクローズ要求システムコールの処理のフローである。このシステムコールは、セマフォ記述子をパラメータに指定してクローズ要求を行う。まず、第１のＯＳ１２００へのコンテキスト切替えを行い（ステップ５８１０）、当該クローズ要求において指定されたセマフォ記述子に対応するエントリをセマフォ管理表５０００からみつけ、そのエントリにおける第１のＯＳのセマフォオブジェクトハンドル５０３０を用いて、第１のＯＳ１２００のセマフォ通信機能１４２０のクローズ要求のシステムコールを発行し（ステップ５８２０）、第１のＯＳのセマフォオブジェクトハンドル有効フラグ５０４０をＯＦＦにし（ステップ５８３０）、エントリ有効フラグ５０１０をＯＦＦにして当該エントリを無効にして（ステップ５８４０）、第２のＯＳ１３００へのコンテキスト切替えを行い（ステップ５８５０）、リターンする（ステップ５８６０）。

【００４０】以上の図１２から図２０に示した処理により、第１のＯＳ１２００がセマフォ機能を有し、第２のＯＳ１３００がセマフォ機能を有さない時、第１のＯＳ上のプロセス１６００が発行する、第１のＯＳ上のプロセス１６００と第２のＯＳ上のプロセス１７００の間のプロセス間通信におけるセマフォ関連要求を、第１のＯＳ１２００が有するセマフォ機能を用いて処理することが可能となり、そして、第２のＯＳ上のプロセス１７００が発行する、第１のＯＳ上のプロセス１６００と第２のＯＳ上のプロセス１７００の間のプロセス間通信におけるセマフォ関連要求を、第１のＯＳ１２００が有するセマフォ機能を用いて処理することが可能となる。

【００４１】次に、メッセージ通信に関連した処理について説明する。

【００４２】図２１は、メッセージ通信関連のデータ構造を示す図である。メッセージ通信関連のデータ構造はＯＳ間通信処理部１８００の中のメッセージ通信機能１８１０の中にあり、メッセージキュー６０００、メッセージヘッダ６０１０、データバッファ６０２０から成る。メッセージキュー６０００には、メッセージヘッダ６０１０へのポインタ、当該メッセージキューに対するメッセージを待つプロセスへのポインタおよび管理情報があり、メッセージヘッダのリストが連結される。この管理情報には、連結されたリスト上のメッセージ数と総バイト数等がある。メッセージキュー６０００は、ＯＳ間通信処理部１８００の初期処理時に確保され、その数はメッセージキュー管理表６１００のエントリ数と同じになる。メッセージヘッダ６０１０には、次のメッセージヘッダ６０１０へのポインタ、メッセージデータへのポインタおよび管理情報が格納されている。この管理情報には、メッセージデータのバイト数等がある。データバッファ６０２０には、メッセージデータが格納される。即ち、メッセージの送り手のメッセージデータがデータバッファ６０２０にコピーされ、更に、メッセージの受け手にコピーされる。

【００４３】図２２は、管理表１９００の中のメッセージキュー管理表６１００を示す図である。メッセージ管理表６１００は、ＯＳ間通信処理部１８００の初期処理時に確保され、そのエントリ数はメッセージキュー６０００の数と同じであり、各エントリはメッセージキューに対応する。メッセージ管理表６１００は、メッセージキュー使用中フラグ６１１０、メッセージキューの名前６１２０、メッセージキューポインタ６１３０から成る。メッセージキュー使用中フラグ６１１０は、このエントリに対応するメッセージキューの使用中を示すフラグである。メッセージキューの名前６１２０は、ＯＳ間通信オブジェクトであるところのメッセージキューの名前が格納される。メッセージキューポインタ６１３０は、このエントリに対応するメッセージキュー６０００へのポインタである。このメッセージキューポインタ６１３０は、メッセージ管理表６１００およびメッセージキュー６０００の確保時に設定される。

【００４４】図２３は、第１のＯＳのプロセス１６００が発行したメッセージ通信のオープン要求システムコールの処理のフローである。このシステムコールは、メッセージキューの名前をパラメータに指定してオープン要求を行い、その結果としてメッセージキュー記述子を得る。以降のメッセージ通信関連のシステムコールでは、このメッセージキュー記述子を用いることになる。まず、メッセージキュー管理表６１００を調べて、メッセージキュー使用中フラグ６１１０がＯＮで、かつ、指定されたメッセージキューの名前とメッセージキューの名前６１２０が同じであるエントリを探す（ステップ６２１０）。メッセージキュー管理表６１００にこのようなエントリがあれば、既にこの名前のメッセージキューはオープンされていることになる。そこで、そのエントリのメッセージキュー記述子を返して終了する（ステップ６２７０）。

【００４５】一方、ステップ６２１０で、メッセージキュー使用中フラグ６１１０がＯＮで、かつ、指定されたメッセージキューの名前とメッセージキューの名前６１２０が同じであるエントリが、メッセージキュー管理表６１００になければ、メッセージキュー使用中フラグ６１１０がＯＦＦのエントリを探す（ステップ６２２０）。なければ、メッセージキュー管理表６１００のすべてのエントリが使用中、即ち、すべてのメッセージキューが使用中であることになるのでエラーリターンする（ステップ６２３０）。メッセージキュー使用中フラグ６１１０がＯＦＦのエントリがあれば、そのエントリのメッセージキュー使用中フラグ６１１０をＯＮにして（ステップ６２４０）、パラメータで指定されたメッセージキューの名前をメッセージキューの名前６１２０に

格納し（ステップ６２５０）、そのエントリのメッセージキュー記述子を返して終了する（ステップ６２６０）。

【００４６】図２４は、第１のＯＳのプロセス１６００が発行したメッセージ通信のメッセージ送信要求システムコールの処理のフローである。このシステムコールは、メッセージキュー記述子、メッセージが格納される領域のアドレス、送信するメッセージのバイト数をパラメータに指定してメッセージ送信要求を行う。まず、送信メッセージのバイト数が上限を超えるか否かをチェックし（ステップ６３１０）、超えればエラーリターンする（ステップ６３９０）。超えなければ、メッセージヘッダ６０１０およびデータバッファ６０２０の両者の割り当てが可能か否かをチェックし（ステップ６３２０）、可能でなければエラーリターンする（ステップ６３８０）。両者とも割り当て可能であれば、割り当てたデータバッファ６０２０に送信メッセージをコピーし（ステップ６３３０）、割り当てたメッセージヘッダ６０１０をパラメータで指定されたメッセージキュー６０００のリストの最後に連結し、当該メッセージキュー６０００の管理情報を更新する（ステップ６３４０）。次に、図４に示したイベント同期通信のオープン要求システムコールおよび図６に示したイベント同期通信のポスト要求システムコールを発行し、当該メッセージキューへのメッセージ到着を待つプロセスを起こし（ステップ６３６０）、リターンする（ステップ６３７０）。

【００４７】図２５は、第１のＯＳのプロセス１６００が発行したメッセージ通信のメッセージ受信要求システムコールの処理のフローである。このシステムコールは、メッセージキュー記述子、メッセージ格納領域アドレスをパラメータに指定してメッセージ受信要求を行う。まず、パラメータで指定されたメッセージキュー６０００にメッセージがあるか否かをチェックする（ステップ６４１０）。あれば、ステップ６４３０へ行く。なければ、図４に示したイベント同期通信のオープン要求システムコールおよび図５に示したイベント同期通信のウェイト要求システムコールを発行し、当該メッセージキューにメッセージが届くのを待つ（ステップ６４２０）。そして、メッセージキュー６０００に連結されたリストの最初のメッセージヘッダ６０１０からポイントされるメッセージバッファ６０２０のメッセージを、パラメータで指定されたメッセージ格納領域アドレスにコピーし（ステップ６４３０）、当該メッセージヘッダ６０１０を当該メッセージキュー６０００のリストから外し、当該メッセージキュー６０００の管理情報を更新し（ステップ６４４０）、リターンする（ステップ６４５０）。

【００４８】図２６は、第１のＯＳのプロセス１６００が発行したメッセージ通信のクローズ要求システムコールの処理のフローである。このシステムコールは、メッセージキュー記述子をパラメータに指定してクローズ要求を行う。パラメータで指定されたメッセージキュー６０００にメッセージがあるか否かをチェックする（ステップ６５１０）。なければ、ステップ６５３０へ行く。あれば、図４に示したイベント同期通信のオープン要求システムコールおよび図６に示したイベント同期通信のポスト要求システムコールを発行し、当該メッセージキューへのメッセージ到着を待つプロセスを起こす（ステップ６５２０）。そして、メッセージキュー使用中フラグ６１１０をＯＦＦにして（ステップ６５３０）、リターンする（ステップ６５４０）。

【００４９】図２７は、第２のＯＳのプロセス１７００が発行したメッセージ通信のオープン要求システムコールの処理のフローである。このシステムコールは、メッセージキューの名前をパラメータに指定してオープン要求を行い、その結果としてメッセージキュー記述子を得る。以降のメッセージ通信関連のシステムコールでは、このメッセージキュー記述子を用いることになる。まず、メッセージキュー管理表６１００を調べて、メッセージキュー使用中フラグ６１１０がＯＮで、かつ、指定されたメッセージキューの名前とメッセージキューの名前６１２０が同じであるエントリを探す（ステップ６６１０）。メッセージキュー管理表６１００にこのようなエントリがあれば、既にこの名前のメッセージキューはオープンされていることになる。そこで、そのエントリのメッセージキュー記述子を返して終了する（ステップ６６７０）。

【００５０】一方、ステップ６２１０で、メッセージキュー使用中フラグ６１１０がＯＮで、かつ、指定されたメッセージキューの名前とメッセージキューの名前６１２０が同じであるエントリが、メッセージキュー管理表６１００になければ、メッセージキュー使用中フラグ６１１０がＯＦＦのエントリを探す（ステップ６６２０）。なければ、メッセージキュー管理表６１００のすべてのエントリが使用中、即ち、すべてのメッセージキューが使用中であることになるのでエラーリターンする（ステップ６６３０）。メッセージキュー使用中フラグ６１１０がＯＦＦのエントリがあれば、そのエントリのメッセージキュー使用中フラグ６１１０をＯＮにして（ステップ６６４０）、パラメータで指定されたメッセージキューの名前をメッセージキューの名前６１２０に格納し（ステップ６６５０）、そのエントリのメッセージキュー記述子を返して終了する（ステップ６６６０）。

【００５１】図２８は、第２のＯＳのプロセス１７００が発行したメッセージ通信のメッセージ送信要求システムコールの処理のフローである。このシステムコールは、メッセージキュー記述子、メッセージが格納される領域のアドレス、送信するメッセージのバイト数をパラメータに指定してメッセージ送信要求を行う。まず、送

信メッセージのバイト数が上限を超えるか否かをチェックし（ステップ６７１０）、超えればエラーリターンする（ステップ６７９０）。超えなければ、メッセージヘッダ６０１０およびデータバッファ６０２０の両者の割り当てが可能か否かをチェックし（ステップ６７２０）、可能でなければエラーリターンする（ステップ６７８０）。両者とも割り当て可能であれば、割り当てたデータバッファ６０２０に送信メッセージをコピーし（ステップ６７３０）、割り当てたメッセージヘッダ６０１０をパラメータで指定されたメッセージキュー６０００のリストの最後に連結し、当該メッセージキュー６０００の管理情報を更新する（ステップ６７４０）。次に、図８に示したイベント同期通信のオープン要求システムコールおよび図１０に示したイベント同期通信のポスト要求システムコールを発行し、当該メッセージキューへのメッセージ到着を待つプロセスを起こし（ステップ６７６０）、リターンする（ステップ６７７０）。

【００５２】図２９は、第２のＯＳのプロセス１７００が発行したメッセージ通信のメッセージ受信要求システムコールの処理のフローである。このシステムコールは、メッセージキュー記述子、メッセージ格納領域アドレスをパラメータに指定してメッセージ受信要求を行う。まず、パラメータで指定されたメッセージキュー６０００にメッセージがあるか否かをチェックする（ステップ６８１０）。あれば、ステップ６８３０へ行く。なければ、図８に示したイベント同期通信のオープン要求システムコールおよび図９に示したイベント同期通信のウェイト要求システムコールを発行し、当該メッセージキューにメッセージが届くのを待つ（ステップ６８２０）。そして、メッセージキュー６０００に連結されたリストの最初のメッセージヘッダ６０１０からポイントされるメッセージバッファ６０２０のメッセージを、パラメータで指定されたメッセージ格納領域アドレスにコピーし（ステップ６８３０）、当該メッセージヘッダ６０１０を当該メッセージキュー６０００のリストから外し、当該メッセージキュー６０００の管理情報を更新し（ステップ６８４０）、リターンする（ステップ６８５０）。

【００５３】図３０は、第２のＯＳのプロセス１７００が発行したメッセージ通信のクローズ要求システムコールの処理のフローである。このシステムコールは、メッセージキュー記述子をパラメータに指定してクローズ要求を行う。パラメータで指定されたメッセージキュー６０００にメッセージがあるか否かをチェックする（ステップ６９１０）。なければ、ステップ６９３０へ行く。あれば、図８に示したイベント同期通信のオープン要求システムコールおよび図１０に示したイベント同期通信のポスト要求システムコールを発行し、当該メッセージキューへのメッセージ到着を待つプロセスを起こす（ステップ６９２０）。そして、メッセージキュー使用中フラグ６１１０をＯＦＦにして（ステップ６９３０）、リターンする（ステップ６９４０）。

【００５４】以上の図２１から図３０に示した処理により、第１のＯＳがメッセージ機能を有さず、第２のＯＳもメッセージ機能を有さない時、ＯＳ間通信処理部１８００上に設けたメッセージキュー６０００の介して、第１のＯＳ上のプロセス１６００と第２のＯＳ上のプロセス１７００の間のプロセス間通信が可能となる。

【００５５】次に、共有メモリに関連した処理について説明する。

【００５６】図３１は、管理表１９００の中の共有メモリ管理表７０００を示す図である。共有メモリ管理表７０００は、ＯＳ間通信処理部１８００の初期処理時に確保される。共有メモリ管理表７０００は、エントリ有効フラグ７０１０、共有メモリの名前７０２０、第１のＯＳの共有メモリオブジェクトハンドル７０３０、第１のＯＳの共有メモリオブジェクトハンドル有効フラグ７０４０から成る。この場合は、第１のＯＳ１２００が共有メモリ機能を有するので、第１のＯＳ１２００の共有メモリ機能のオブジェクトハンドルを格納するのが共有メモリオブジェクトハンドル７０３０であり、共有メモリオブジェクトハンドル有効フラグ７０４０はそれの有効か無効かを示すフラグである。また、エントリ有効フラグ７０１０は、このエントリが有効か無効かを示すフラグである。共有メモリの名前７０２０は、ＯＳ間通信オブジェクトであるところの共有メモリの名前を格納する。

【００５７】図３２は、第１のＯＳのプロセス１６００が発行した共有メモリ通信のオープン要求システムコールの処理のフローである。このシステムコールは、共有メモリの名前をパラメータに指定してオープン要求を行い、その結果として共有メモリ記述子を得る。以降の共有メモリ通信関連のシステムコールでは、この共有メモリ記述子を用いることになる。第１のＯＳ１２００は共有メモリ通信機能を有する。まず、共有メモリ管理表７０００を調べて、エントリ有効フラグ７０１０がＯＮで、かつ、指定された共有メモリの名前と共有メモリの名前７０２０が同じであるエントリを探す（ステップ７１１０）。共有メモリ管理表７０００にこのようなエントリがあれば、既にこの名前のセマフォはオープンされていることになる。そこで、そのエントリの共有メモリ記述子を返して終了する（ステップ７１９０）。

【００５８】一方、ステップ７１１０で、エントリ有効フラグ７０１０がＯＮで、かつ、指定された共有メモリの名前と共有メモリの名前７０２０が同じであるエントリが、セマフォ管理表７０００になければ、エントリ有効フラグ７０１０がＯＦＦのエントリを探す（ステップ７１２０）。なければ、共有メモリ管理表７０００のすべてのエントリが使用中であることになるのでエラーリターンする（ステップ７１３０）。エントリ有効フラグ

７０１０がＯＦＦのエントリがあれば、第１のＯＳ１２００の共有メモリ通信機能１４４０のオープンのシステムコールをこの共有メモリの名前で発行することにより、第１のＯＳ１２００の共有メモリ通信機能１４４０のオブジェクトハンドルを獲得し（ステップ７１４０）、獲得したオブジェクトハンドルを第１のＯＳの共有メモリオブジェクトハンドル７０３０にセットし（ステップ７１５０）、第１のＯＳの共有メモリオブジェクトハンドル有効フラグ７０４０をＯＮにし（ステップ７１６０）、パラメータで指定された共有メモリの名前を共有メモリの名前７０２０に格納し（ステップ７１６５）、エントリ有効フラグ７０１０をＯＮにして（ステップ７１７０）、そのエントリの共有メモリ記述子を返して終了する（ステップ７１８０）。

【００５９】図３３は、第１のＯＳのプロセス１６００が発行した共有メモリ通信の仮想空間マッピング要求システムコールの処理のフローである。このシステムコールは、共有メモリ記述子をパラメータに指定して仮想空間マッピング要求を行い、マッピングされた仮想空間アドレスを得る。まず、当該仮想空間マッピング要求において指定された共有メモリ記述子に対応するエントリを共有メモリ管理表７０００からみつけ、そのエントリにおける第１のＯＳの共有メモリオブジェクトハンドル７０３０を用いて、第１のＯＳ１２００の共有メモリ通信機能１４４０の仮想空間マッピング要求のシステムコールを発行し（ステップ７２１０）、マッピングされた仮想空間アドレスをリターンする（ステップ７２２０）。

【００６０】図３４は、第１のＯＳのプロセス１６００が発行した共有メモリ通信の仮想空間マッピング解除要求システムコールの処理のフローである。このシステムコールは、共有メモリ記述子をパラメータに指定して仮想空間マッピング解除要求を行う。まず、当該仮想空間マッピング解除要求において指定された共有メモリ記述子に対応するエントリを共有メモリ管理表７０００からみつけ、そのエントリにおける第１のＯＳの共有メモリオブジェクトハンドル７０３０を用いて、第１のＯＳ１２００の共有メモリ通信機能１４４０の仮想空間マッピング解除要求のシステムコールを発行し（ステップ７３１０）、リターンする（ステップ７３２０）。

【００６１】図３５は、第１のＯＳのプロセス１６００が発行した共有メモリ通信のクローズ要求システムコールの処理のフローである。このシステムコールは、共有メモリ記述子をパラメータに指定してクローズ要求を行う。まず、当該クローズ要求において指定された共有メモリ記述子に対応するエントリを共有メモリ管理表７０００からみつけ、そのエントリにおける第１のＯＳの共有メモリオブジェクトハンドル７０３０を用いて、第１のＯＳ１２００の共有メモリ通信機能１４４０のクローズ要求のシステムコールを発行し（ステップ７４１０）、第１のＯＳの共有メモリオブジェクトハンドル有効フラグ７０４０をＯＦＦにし（ステップ７４２０）、エントリ有効フラグ７０１０をＯＦＦにして当該エントリを無効にして（ステップ７４３０）、リターンする（ステップ７４４０）。

【００６２】図３６は、第２のＯＳのプロセス１７００が発行した共有メモリ通信のオープン要求システムコールの処理のフローである。このシステムコールは、共有メモリの名前をパラメータに指定してオープン要求を行い、その結果として共有メモリ記述子を得る。以降の共有メモリ通信関連のシステムコールでは、この共有メモリ記述子を用いることになる。第２のＯＳ１３００は共有メモリ通信機能を有しない。まず、共有メモリ管理表７０００を調べて、エントリ有効フラグ７０１０がＯＮで、かつ、指定された共有メモリの名前と共有メモリの名前７０２０が同じであるエントリを探す（ステップ７５１０）。共有メモリ管理表７０００にこのようなエントリがあれば、既にこの名前の共有メモリはオープンされていることになる。そこで、そのエントリの共有メモリ記述子を返して終了する（ステップ７５９０）。

【００６３】一方、ステップ７５１０で、エントリ有効フラグ７０１０がＯＮで、かつ、指定された共有メモリの名前と共有メモリの名前７０２０が同じであるエントリが、共有メモリ管理表７０００になければ、エントリ有効フラグ７０１０がＯＦＦのエントリを探す（ステップ７５２０）。なければ、共有メモリ管理表７０００のすべてのエントリが使用中であることになるのでエラーリターンする（ステップ７５３０）。エントリ有効フラグ７０１０がＯＦＦのエントリがあれば、第１のＯＳ１２００へコンテキスト切替えを行い（ステップ７５３５）、第１のＯＳ１２００の共有メモリ通信機能１４４０のオープンのシステムコールをこの共有メモリの名前で発行することにより、第１のＯＳ１２００の共有メモリ通信機能１４４０のオブジェクトハンドルを獲得し（ステップ７５４０）、獲得したオブジェクトハンドルを第１のＯＳの共有メモリオブジェクトハンドル７０３０にセットし（ステップ７５５０）、第１のＯＳの共有メモリオブジェクトハンドル有効フラグ７０４０をＯＮにし（ステップ７５６０）、パラメータとして指定された共有メモリの名前を共有メモリの名前７０２０に格納し（ステップ７５６５）、エントリ有効フラグ７０１０をＯＮにして（ステップ７５７０）、第２のＯＳ１３００へコンテキスト切替えを行い（ステップ７５７５）、そのエントリの共有メモリ記述子を返して終了する（ステップ７５８０）。

【００６４】図３７は、第２のＯＳのプロセス１７００が発行した共有メモリ通信の仮想空間マッピング要求システムコールの処理のフローである。このシステムコールは、共有メモリ記述子をパラメータに指定して仮想空間マッピング要求を行う。まず、第１のＯＳ１２００へのコンテキスト切替えを行い（ステップ７６１０）、当

該仮想空間マッピング要求において指定された共有メモリ記述子に対応するエントリを共有メモリ管理表７０００からみつけ、そのエントリにおける第１のＯＳの共有メモリオブジェクトハンドル７０３０を用いて、第１のＯＳ１２００の共有メモリ通信機能１４４０の仮想空間マッピング要求のシステムコールを発行し（ステップ７６２０）、第２のＯＳ１３００へのコンテキスト切替えを行い（ステップ７６３０）、リターンする（ステップ７６４０）。

【００６５】図３８は、第２のＯＳのプロセス１７００が発行した共有メモリ通信の仮想空間マッピング解除要求システムコールの処理のフローである。このシステムコールは、共有メモリ記述子をパラメータに指定して仮想空間マッピング解除要求を行う。まず、第１のＯＳ１２００へのコンテキスト切替えを行い（ステップ７７１０）、当該仮想空間マッピング要求において指定された共有メモリ記述子に対応するエントリを共有メモリ管理表７０００からみつけ、そのエントリにおける第１のＯＳの共有メモリオブジェクトハンドル７０３０を用いて、第１のＯＳ１２００の共有メモリ通信機能１４４０の仮想空間マッピング解除要求のシステムコールを発行し（ステップ７７２０）、第２のＯＳ１３００へのコンテキスト切替えを行い（ステップ７７３０）、リターンする（ステップ７７４０）。

【００６６】図３９は、第２のＯＳのプロセス１７００が発行した共有メモリ通信のクローズ要求システムコールの処理のフローである。このシステムコールは、共有メモリ記述子をパラメータに指定してクローズ要求を行う。まず、第１のＯＳ１２００へのコンテキスト切替えを行い（ステップ７８１０）、当該クローズ要求において指定された共有メモリ記述子に対応するエントリを共有メモリ管理表７０００からみつけ、そのエントリにおける第１のＯＳの共有メモリオブジェクトハンドル７０３０を用いて、第１のＯＳ１２００の共有メモリ通信機能１４４０のクローズ要求のシステムコールを発行し（ステップ７８２０）、第１のＯＳの共有メモリオブジェクトハンドル有効フラグ７０４０をＯＦＦにし（ステップ７８３０）、エントリ有効フラグ７０１０をＯＦＦにして当該エントリを無効にして（ステップ７８４０）、第２のＯＳ１３００へのコンテキスト切替えを行い（ステップ７８５０）、リターンする（ステップ７８６０）。

【００６７】以上の図３１から図３９に示した処理により、第１のＯＳ１２００が共有メモリ機能を有し、第２のＯＳ１３００が共有メモリ機能を有さない時、第１のＯＳ上のプロセス１６００が発行する、第１のＯＳ上のプロセス１６００と第２のＯＳ上のプロセス１７００の間のプロセス間通信における共有メモリ関連要求を、第１のＯＳ１２００が有する共有メモリ機能を用いて処理することが可能となり、そして、第２のＯＳ上のプロセス１７００が発行する、第１のＯＳ上のプロセス１６００と第２のＯＳ上のプロセス１７００の間のプロセス間通信における共有メモリ関連要求を、第１のＯＳ１２００が有する共有メモリ機能を用いて処理することが可能となる。

【００６８】

【発明の効果】１台の計算機上で複数のオペレーティングシステム（ＯＳ）が動作するとき、それぞれのＯＳ上で動作するプロセス間でのプロセス間通信（イベント同期、セマフォ、メッセージ、共有メモリ）が可能となる。そして、この通信において、それぞれのＯＳが持つ自身のプロセス間通信機能を利用するので、実現が容易になる。

【図面の簡単な説明】

【図１】本発明の実施例を実施する計算機システムにおけるＯＳ間通信の概要を示す図である。

【図２】第１のＯＳのプロセス１６００および第２のＯＳのプロセス１７００のプロセスのアドレス空間の構成を示す図である。

【図３】管理表１９００の中のイベント管理表３０００を示す図である。

【図４】第１のＯＳのプロセス１６００が発行したイベント同期通信のオープン要求システムコールの処理のフローである。

【図５】第１のＯＳのプロセス１６００が発行したイベント同期通信のウェイト要求システムコールの処理のフローである。

【図６】第１のＯＳのプロセス１６００が発行したイベント同期通信のポスト要求システムコールの処理のフローである。

【図７】第１のＯＳのプロセス１６００が発行したイベント同期通信のクローズ要求システムコールの処理のフローである。

【図８】第２のＯＳのプロセス１７００が発行したイベント同期通信のオープン要求システムコールの処理のフローである。

【図９】第２のＯＳのプロセス１７００が発行したイベント同期通信のウェイト要求システムコールの処理のフローである。

【図１０】第２のＯＳのプロセス１７００が発行したイベント同期通信のポスト要求システムコールの処理のフローである。

【図１１】第２のＯＳのプロセス１７００が発行したイベント同期通信のクローズ要求システムコールの処理のフローである。

【図１２】管理表１９００の中のセマフォ管理表５０００を示す図である。

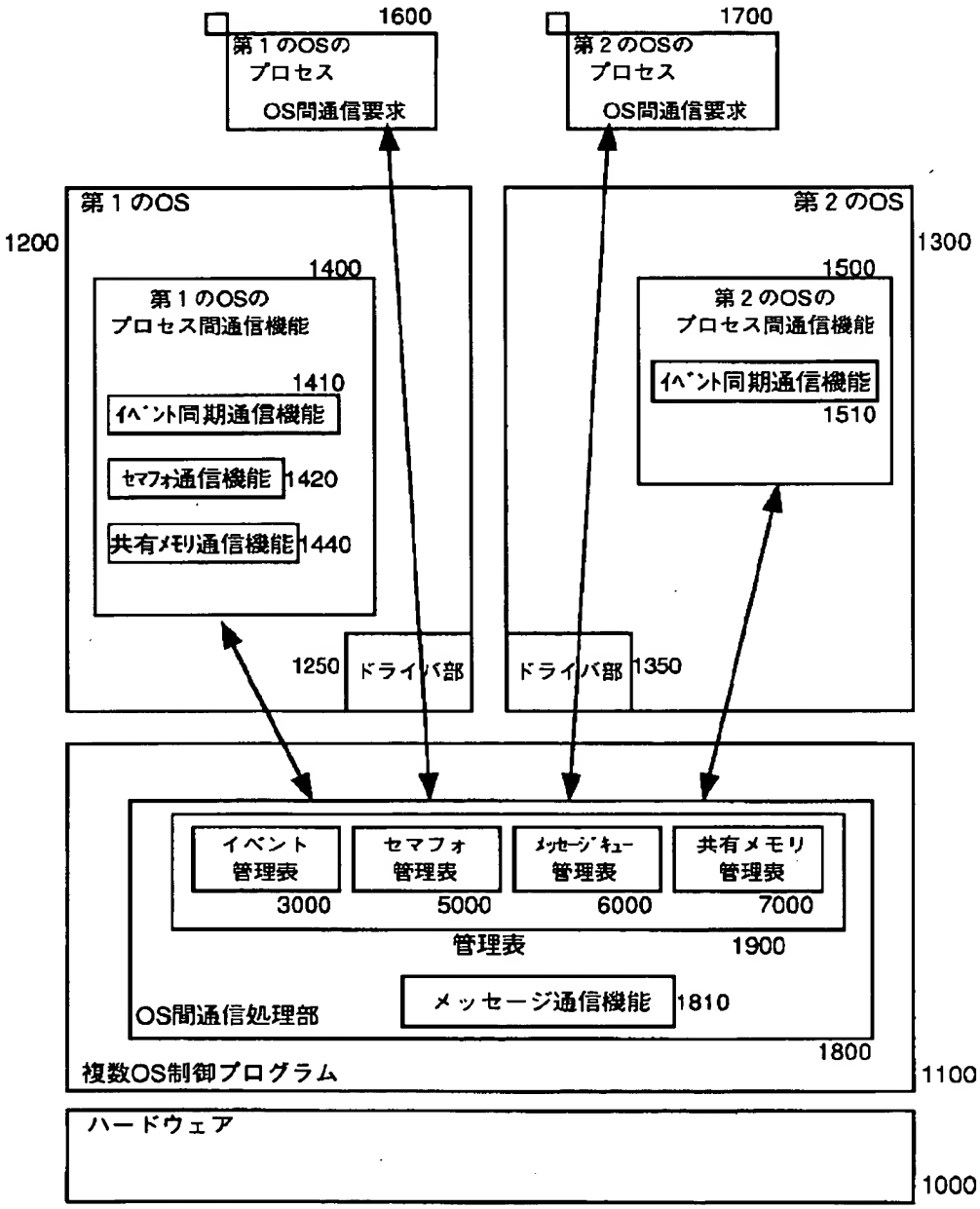
【図１３】第１のＯＳのプロセス１６００が発行したセマフォ通信のオープン要求システムコールの処理のフローである。

【図１４】第１のＯＳのプロセス１６００が発行したセマフォ通信のセマフォウェイト要求システムコールの処理のフローである。
【図１５】第１のＯＳのプロセス１６００が発行したセマフォ通信のセマフォカウントアップ要求システムコールの処理のフローである。
【図１６】第１のＯＳのプロセス１６００が発行したセマフォ通信のクローズ要求システムコールの処理のフローである。
【図１７】第２のＯＳのプロセス１７００が発行したセマフォ通信のオープン要求システムコールの処理のフローである。
【図１８】第２のＯＳのプロセス１７００が発行したセマフォ通信のセマフォウェイト要求システムコールの処理のフローである。
【図１９】第２のＯＳのプロセス１７００が発行したセマフォ通信のセマフォカウントアップ要求システムコールの処理のフローである。
【図２０】第２のＯＳのプロセス１７００が発行したセマフォ通信のクローズ要求システムコールの処理のフローである。
【図２１】メッセージ通信関連のデータ構造を示す図である。
【図２２】管理表１９００の中のメッセージキュー管理表６１００を示す図である。
【図２３】第１のＯＳのプロセス１６００が発行したメッセージ通信のオープン要求システムコールの処理のフローである。
【図２４】第１のＯＳのプロセス１６００が発行したメッセージ通信のメッセージ送信要求システムコールの処理のフローである。
【図２５】第１のＯＳのプロセス１６００が発行したメッセージ通信のメッセージ受信要求システムコールの処理のフローである。
【図２６】第１のＯＳのプロセス１６００が発行したメッセージ通信のクローズ要求システムコールの処理のフローである。
【図２７】第２のＯＳのプロセス１７００が発行したメッセージ通信のオープン要求システムコールの処理のフローである。
【図２８】第２のＯＳのプロセス１７００が発行したメッセージ通信のメッセージ送信要求システムコールの処理のフローである。
【図２９】第２のＯＳのプロセス１７００が発行したメッセージ通信のメッセージ受信要求システムコールの処理のフローである。
【図３０】第２のＯＳのプロセス１７００が発行したメッセージ通信のクローズ要求システムコールの処理のフローである。
【図３１】管理表１９００の中の共有メモリ管理表７００を示す図である。
【図３２】第１のＯＳのプロセス１６００が発行した共有メモリ通信のオープン要求システムコールの処理のフローである。
【図３３】第１のＯＳのプロセス１６００が発行した共有メモリ通信の仮想空間マッピング要求システムコールの処理のフローである。
【図３４】第１のＯＳのプロセス１６００が発行した共有メモリ通信の仮想空間マッピング解除要求システムコールの処理のフローである。
【図３５】第１のＯＳのプロセス１６００が発行した共有メモリ通信のクローズ要求システムコールの処理のフローである。
【図３６】第２のＯＳのプロセス１７００が発行した共有メモリ通信のオープン要求システムコールの処理のフローである。
【図３７】第２のＯＳのプロセス１７００が発行した共有メモリ通信の仮想空間マッピング要求システムコールの処理のフローである。
【図３８】第２のＯＳのプロセス１７００が発行した共有メモリ通信の仮想空間マッピング解除要求システムコールの処理のフローである。
【図３９】第２のＯＳのプロセス１７００が発行した共有メモリ通信のクローズ要求システムコールの処理のフローである。
【符号の説明】
１０００：ハードウェア、１１００：複数ＯＳ制御プログラム、１２００：第１のＯＳ、１２５０：ドライバ部、１３００：第２のＯＳ、１３５０：ドライバ部、１４００：第１のＯＳのプロセス間通信機能、１４１０：イベント同期通信機能、１４２０：セマフォ通信機能、１４４０：共有メモリ通信機能、１５００：第２のＯＳのプロセス間通信機能、１５１０：イベント同期通信機能、１６００：第１のＯＳのプロセス、１７００：第２のＯＳのプロセス、１８００：ＯＳ間通信処理部、１８１０：メッセージ通信機能、１９００：管理表、２１００：第１のＯＳのプロセス１６００のアドレス空間、２２００：第２のＯＳのプロセス１７００のアドレス空間、２３００：アプリケーション領域、２４００：システム領域、３０００：イベント管理表、３０１０：エントリ有効フラグ、３０２０：イベントの名前、３０３０：第１のＯＳのイベントオブジェクトハンドル、３０４０：第１のＯＳのイベントオブジェクトハンドル有効フラグ、３０５０：第２のＯＳのイベントオブジェクトハンドル、３０６０：第２のＯＳのイベントオブジェクトハンドル有効フラグ、５０００：セマフォ管理表、５０１０：エントリ有効フラグ、５０２０：セマフォの名前、５０３０：第１のＯＳのセマフォオブジェクトハンドル、５０４０：第１のＯＳのセマフォオブジェクトハンドル有効フラグ、６０００：メッセージキュー、６０

１０：メッセージヘッダ、６０２０：データバッファ、６１００：メッセージキュー管理表、６１１０：メッセージキュー使用中フラグ６１１０、６１２０：メッセージキューの名前、６１３０：メッセージキューポインタ、７０００：共有メモリ管理表、７０１０：エントリ有効フラグ、７０２０：共有メモリの名前、７０３０：第１のＯＳの共有メモリオブジェクトハンドル、７０４０：第１のＯＳの共有メモリオブジェクトハンドル有効フラグ。

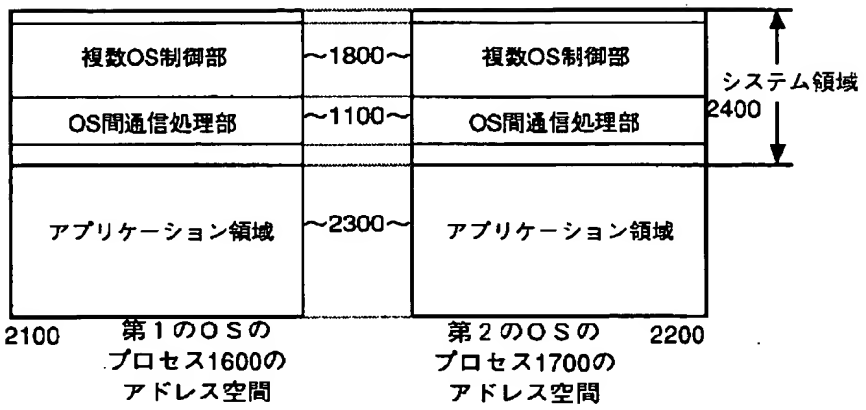
【図１】

図１



【図２】

図２



【図３】

図３

3000

エントリ有効フラグ	イベントの名前	第１のOSのイベントオブジェクトハンドル	第１のOSのイベントオブジェクトハンドル有効フラグ	第２のOSのイベントオブジェクトハンドル	第２のOSのイベントオブジェクトハンドル有効フラグ
3010	3020	3030	3040	3050	3060

【図１２】

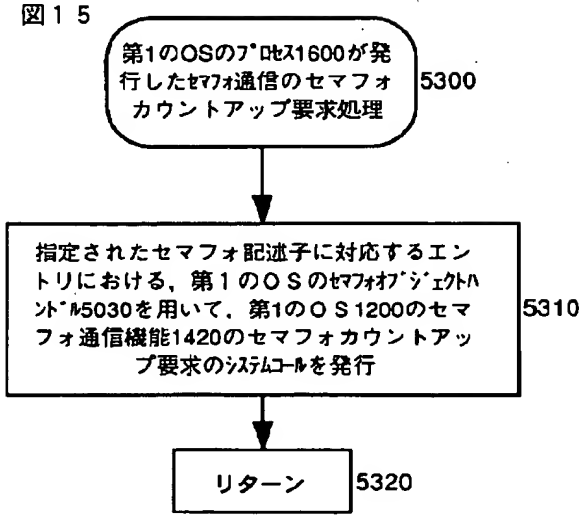
図１２

5000

エントリ有効フラグ	セマフォの名前	第１のOSのセマフォオブジェクトハンドル	第１のOSのセマフォオブジェクトハンドル有効フラグ
5010	5020	5030	5040

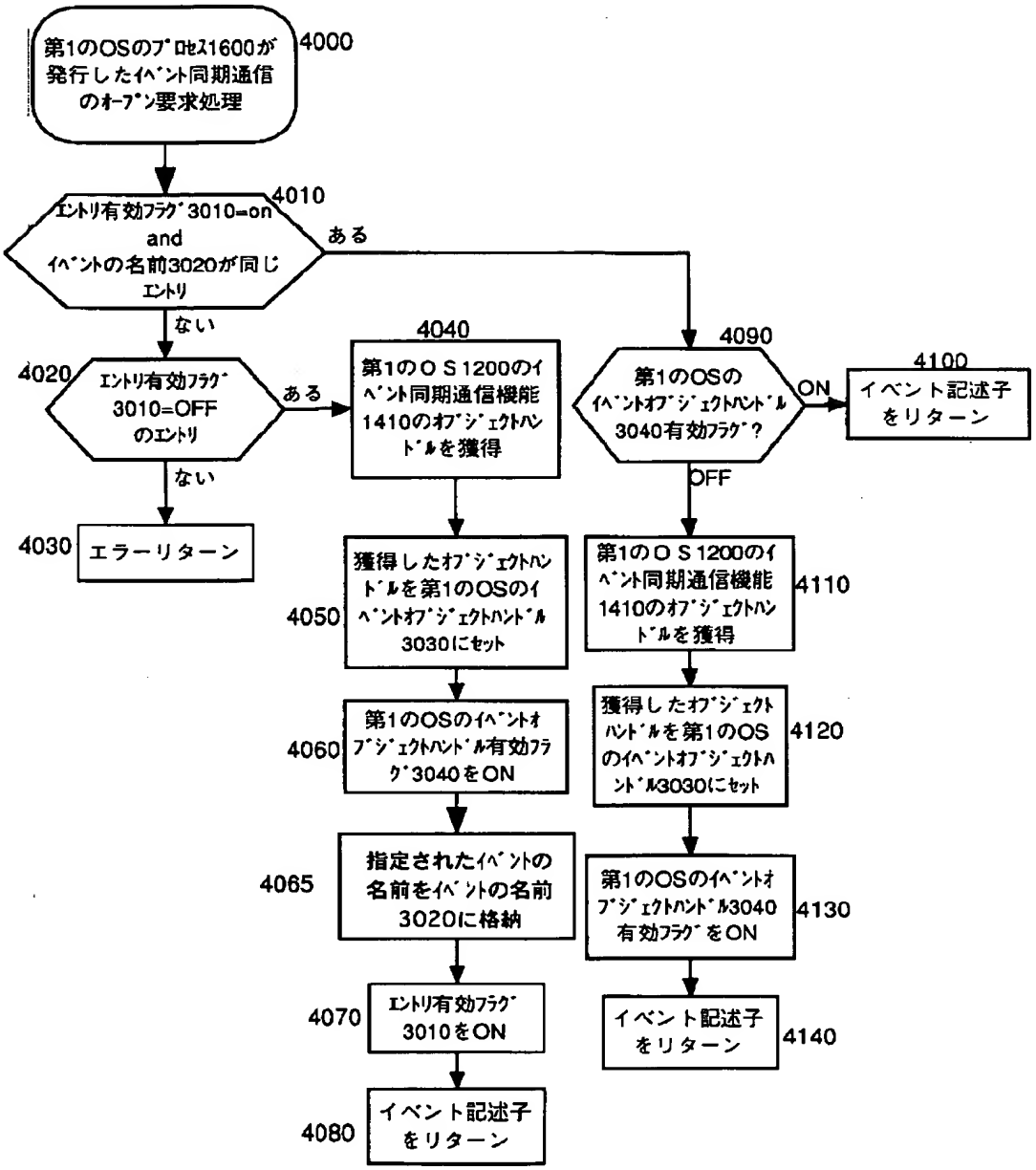
【図１５】

図１５



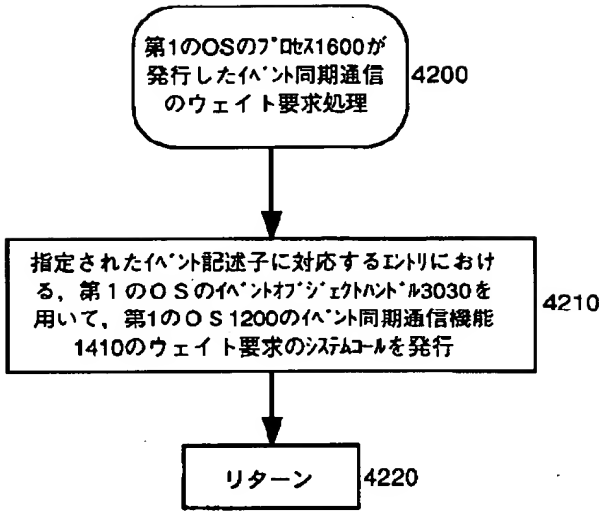
【図４】

図 4


第1のOSのﾌﾟﾛｾｽ1600が発行したｲﾍﾞﾝﾄ同期通信のｵｰﾌﾟﾝ要求処理
4000
ｴﾝﾄﾘ有効ﾌﾗｸﾞ3010=on and ｲﾍﾞﾝﾄの名前3020が同じｴﾝﾄﾘ
4010
ある
ない
4020
ｴﾝﾄﾘ有効ﾌﾗｸﾞ3010=OFFのｴﾝﾄﾘ
ある
ない
4030
エラーリターン
4040
第1のＯＳ1200のｲﾍﾞﾝﾄ同期通信機能1410のｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを獲得
4050
獲得したｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを第1のOSのｲﾍﾞﾝﾄｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3030にｾｯﾄ
4060
第1のOSのｲﾍﾞﾝﾄｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ有効ﾌﾗｸﾞ3040をON
4065
指定されたｲﾍﾞﾝﾄの名前をｲﾍﾞﾝﾄの名前3020に格納
4070
ｴﾝﾄﾘ有効ﾌﾗｸﾞ3010をON
4080
イベント記述子をリターン
4090
第1のOSのｲﾍﾞﾝﾄｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3040有効ﾌﾗｸﾞ?
ON
4100
イベント記述子をリターン
OFF
4110
第1のＯＳ1200のｲﾍﾞﾝﾄ同期通信機能1410のｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを獲得
4120
獲得したｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを第1のOSのｲﾍﾞﾝﾄｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3030にｾｯﾄ
4130
第1のOSのｲﾍﾞﾝﾄｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ3040有効ﾌﾗｸﾞをON
4140
イベント記述子をリターン

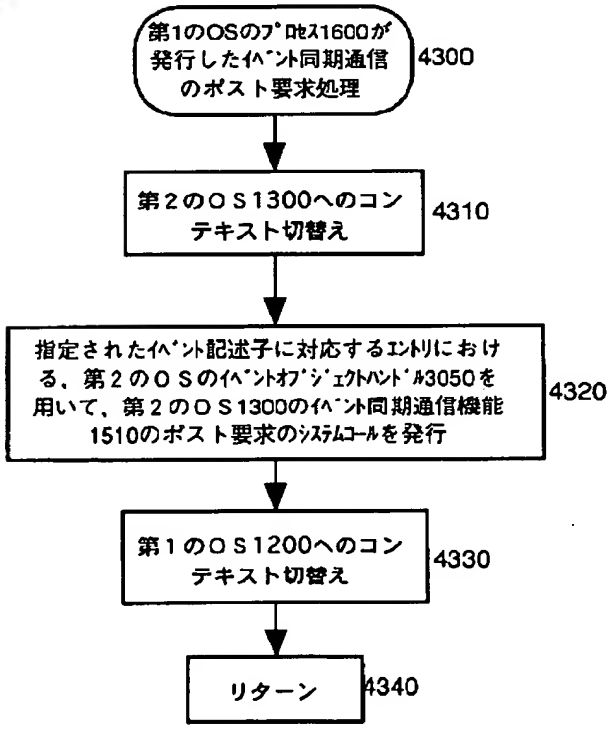
【図５】

図５



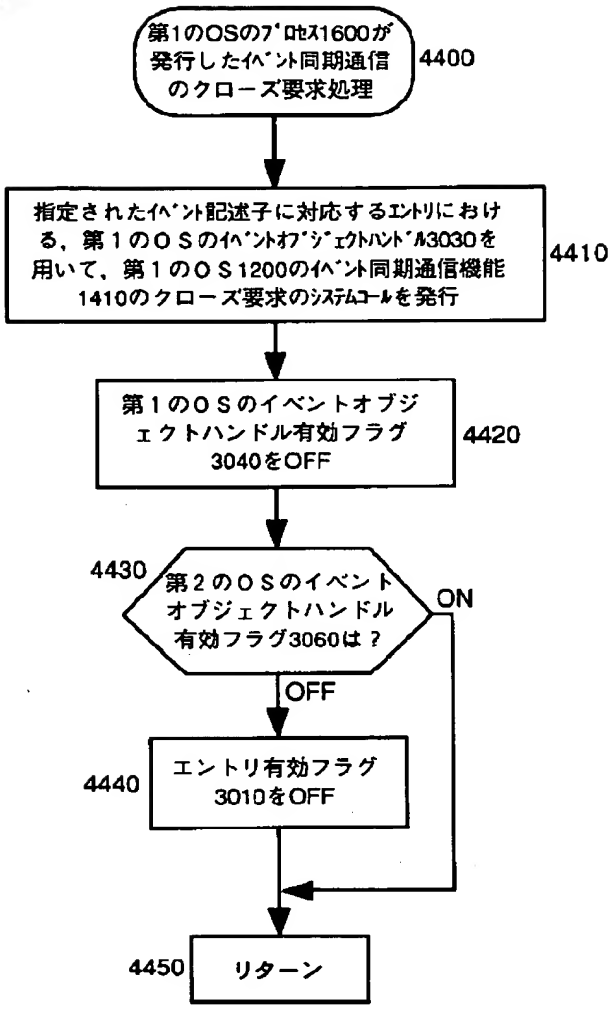
【図６】

図６

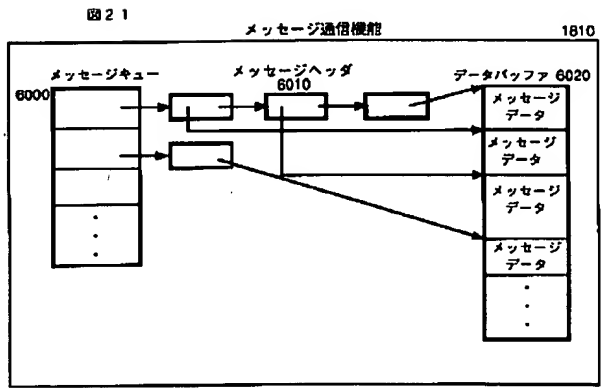


【図７】

図７

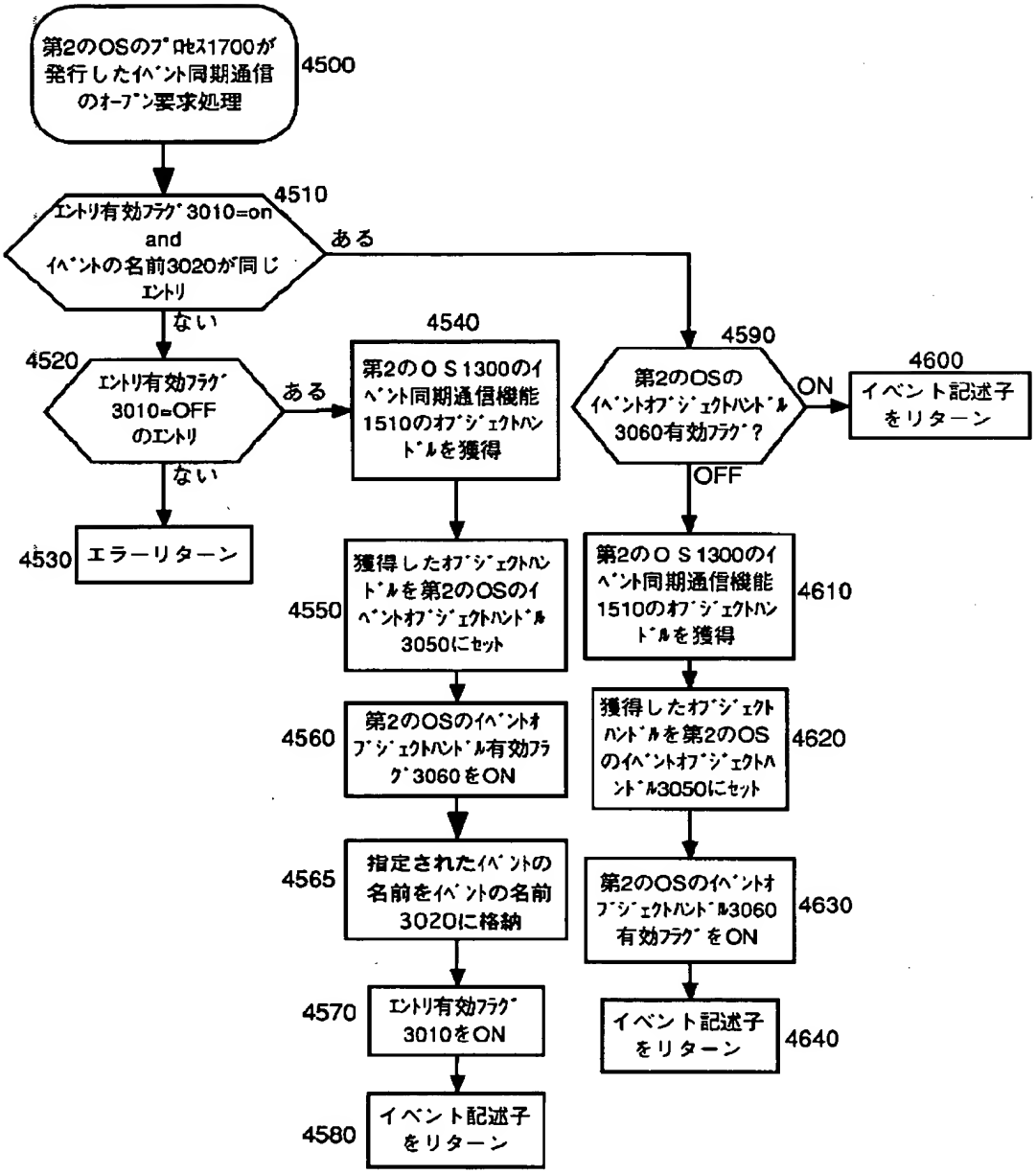


【図２１】



【図８】

図８



【図２２】

図２２

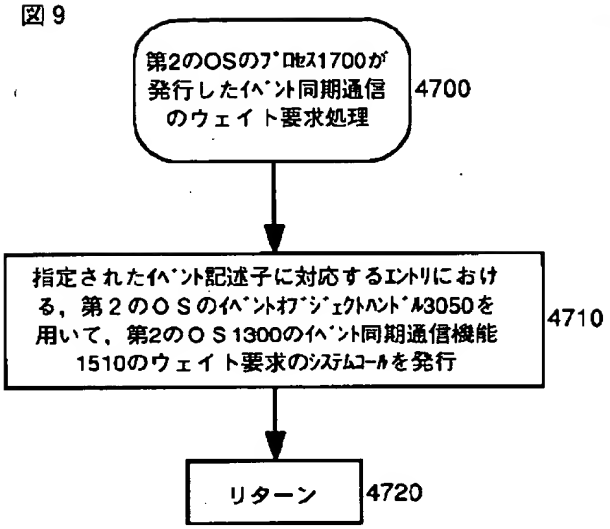
6100

ﾒｯｾｰｼﾞｷｭｰ使用中ﾌﾗｸﾞ	ﾒｯｾｰｼﾞｷｭｰの名前	ﾒｯｾｰｼﾞｷｭｰﾎﾟｲﾝﾀ

6110　6120　6130

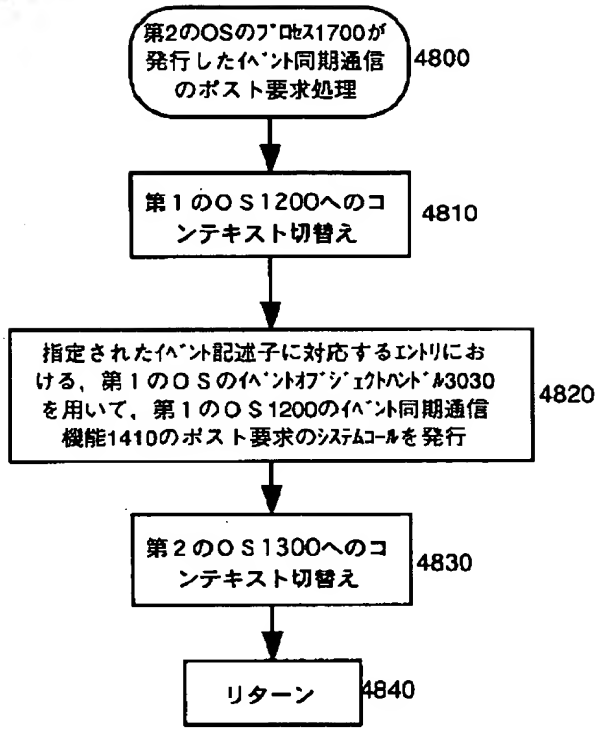
【図９】

図９



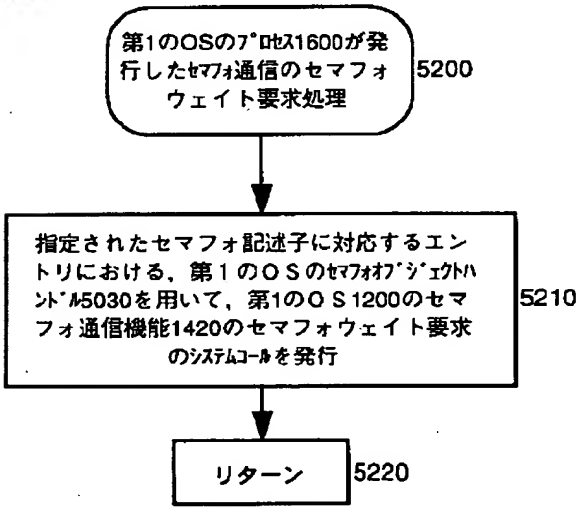
【図１０】

図１０



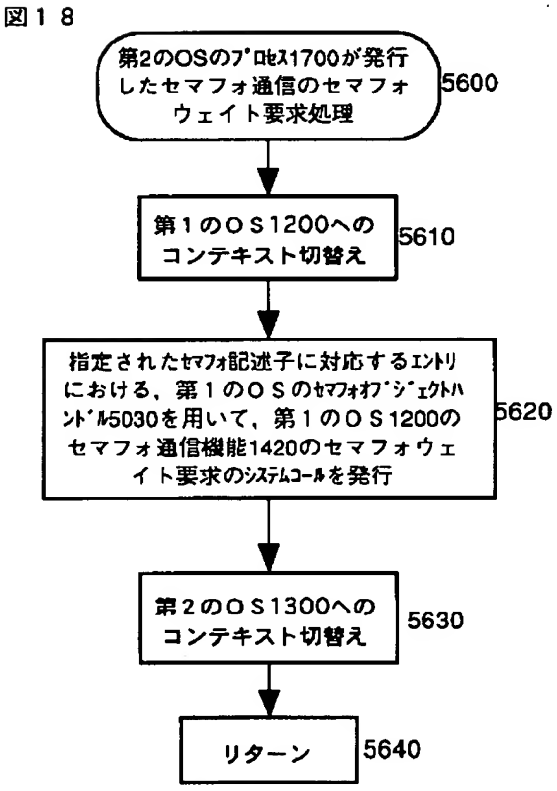
【図１４】

図１４



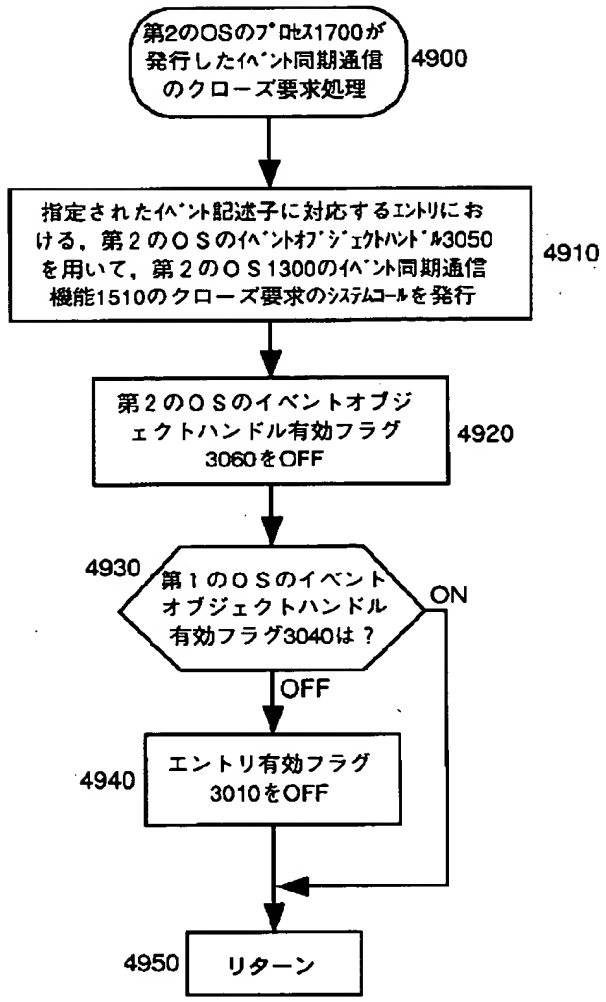
【図１８】

図１８



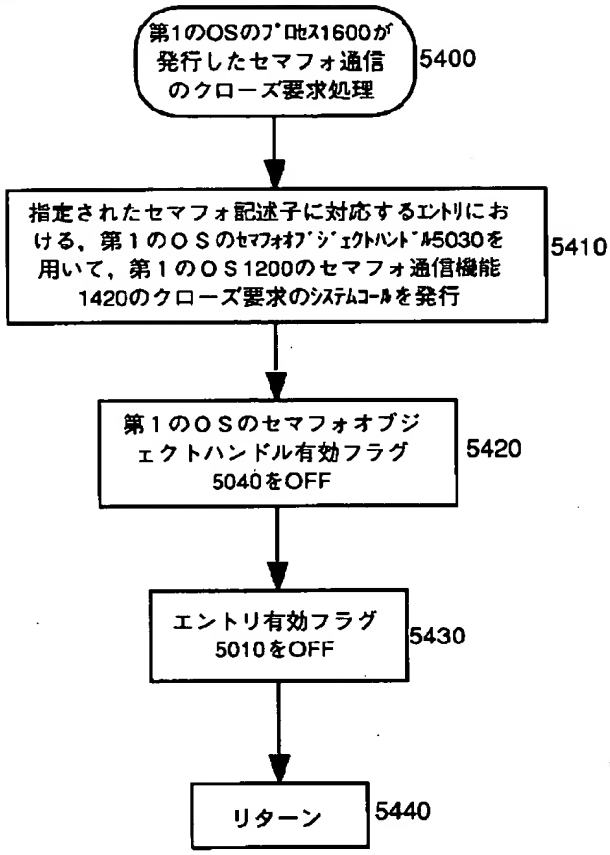
【図１１】

図１１



【図１６】

図１６



【図３１】

図３１

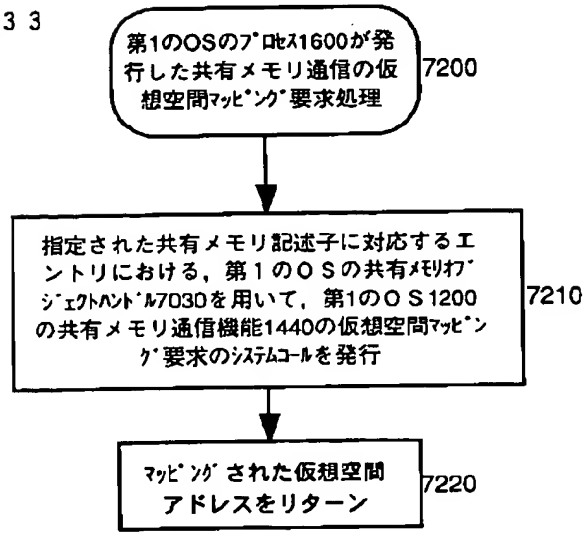
7000

エントリ有効フラグ	共有メモリの名前	第１のOSの共有メモリオブジェクトハンドル	第１のOSの共有メモリオブジェクトハンドル有効フラグ

7010　7020　7030　7040

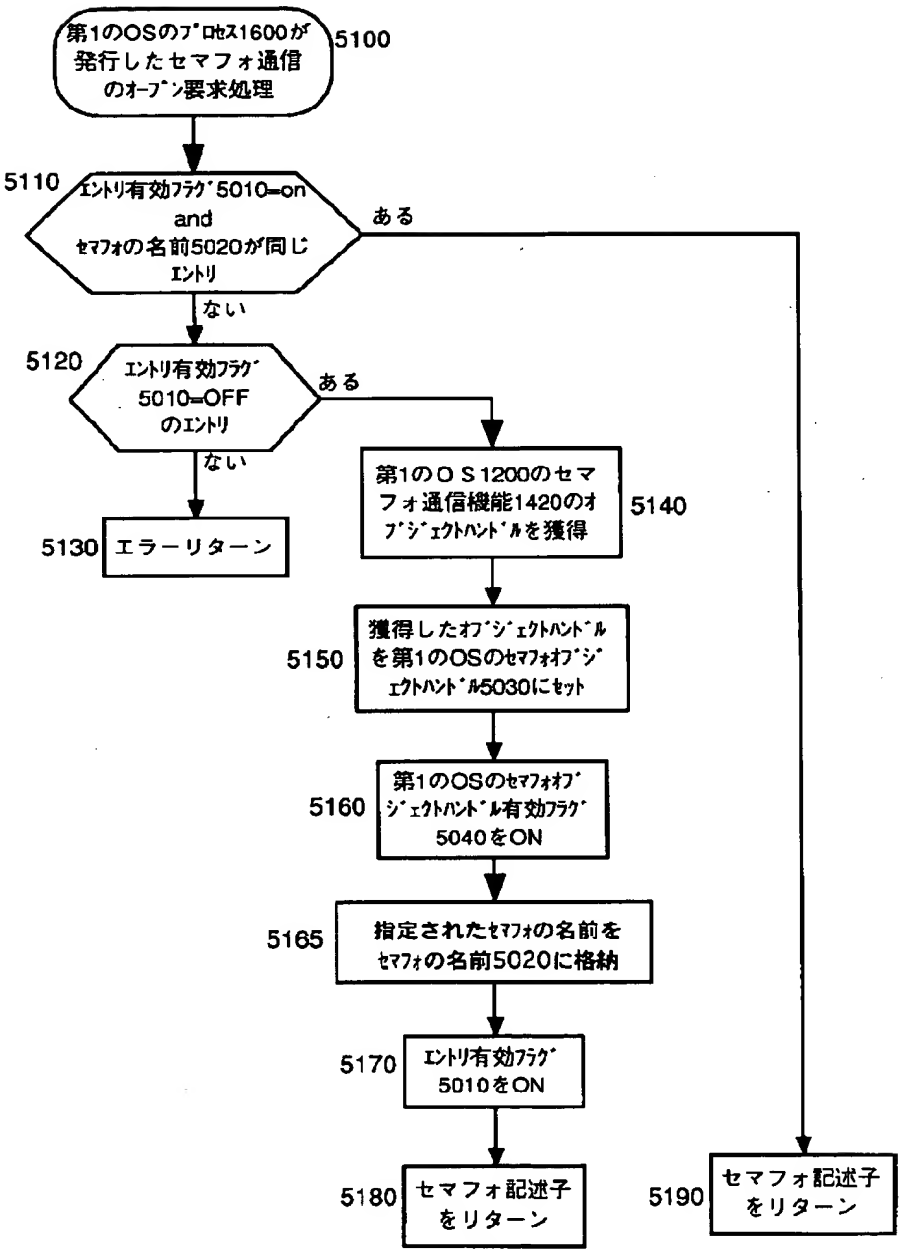
【図３３】

図３３



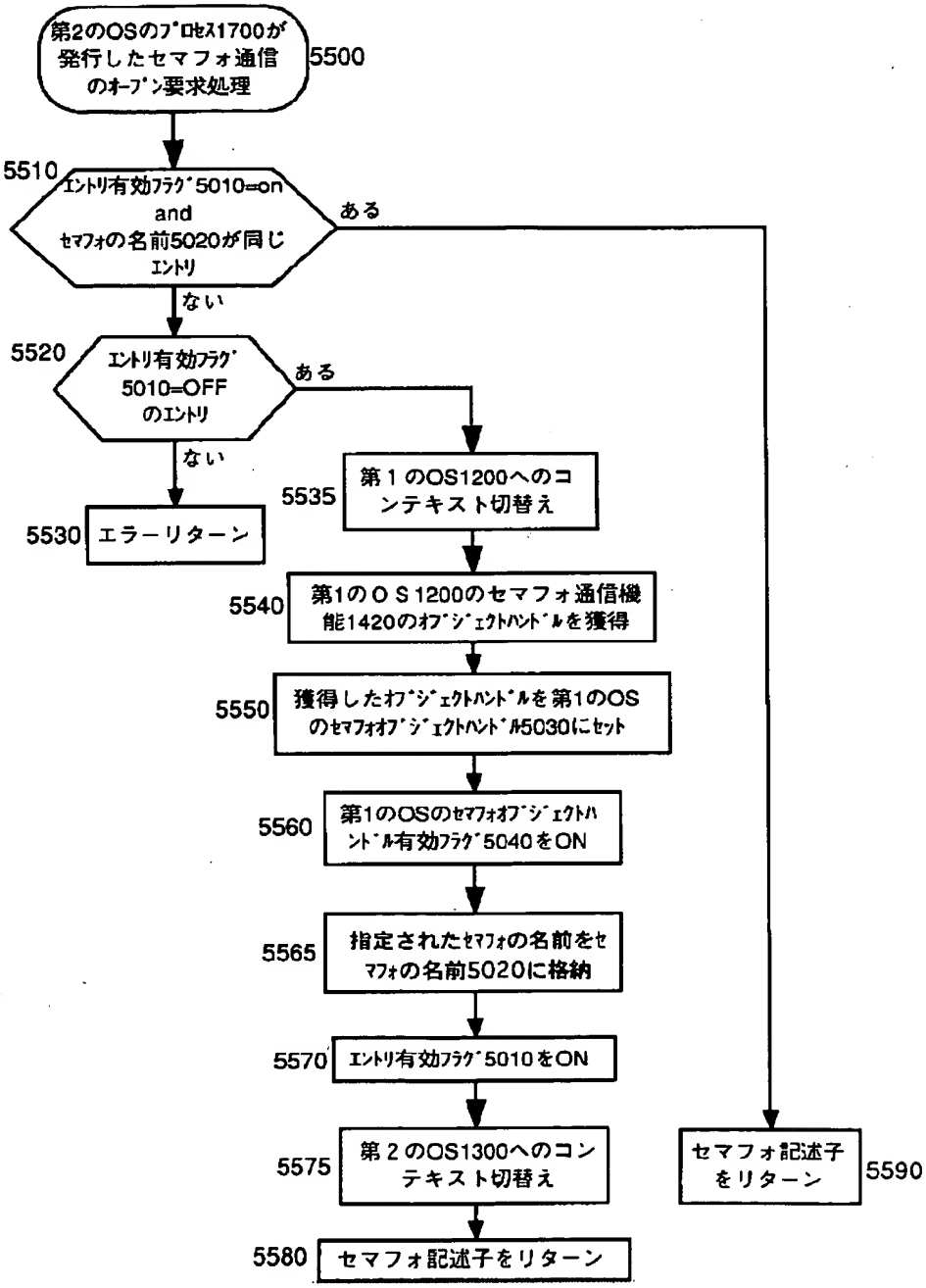
【図１３】

図１３


第1のOSのﾌﾟﾛｾｽ1600が発行したセマフォ通信のｵｰﾌﾟﾝ要求処理
5100
5110
ｴﾝﾄﾘ有効ﾌﾗｸﾞ5010=on and ｾﾏﾌｫの名前5020が同じｴﾝﾄﾘ
ある
ない
5120
ｴﾝﾄﾘ有効ﾌﾗｸﾞ5010=OFFのｴﾝﾄﾘ
ある
ない
5130
エラーリターン
第1のＯＳ1200のセマフォ通信機能1420のｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを獲得
5140
5150
獲得したｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを第1のOSのｾﾏﾌｫｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ5030にｾｯﾄ
5160
第1のOSのｾﾏﾌｫｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ有効ﾌﾗｸﾞ5040をON
5165
指定されたｾﾏﾌｫの名前をｾﾏﾌｫの名前5020に格納
5170
ｴﾝﾄﾘ有効ﾌﾗｸﾞ5010をON
5180
セマフォ記述子をリターン
5190
セマフォ記述子をリターン

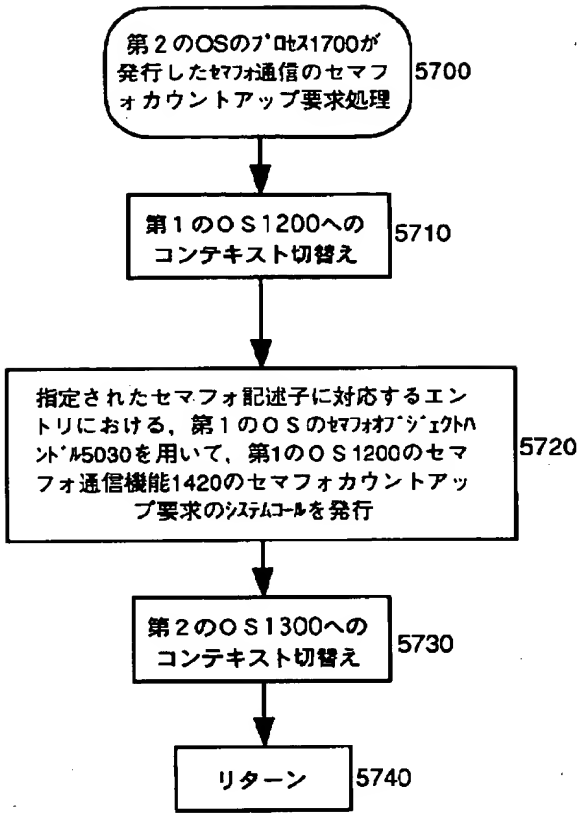
【図１７】

図１７


第2のOSのﾌﾟﾛｾｽ1700が発行したセマフォ通信のｵｰﾌﾟﾝ要求処理
5500
5510
ｴﾝﾄﾘ有効ﾌﾗｸﾞ5010=on and ｾﾏﾌｫの名前5020が同じｴﾝﾄﾘ
ある
ない
5520
ｴﾝﾄﾘ有効ﾌﾗｸﾞ5010=OFFのｴﾝﾄﾘ
ある
ない
5530
エラーリターン
5535
第１のOS1200へのコンテキスト切替え
5540
第1のＯＳ1200のセマフォ通信機能1420のｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを獲得
5550
獲得したｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを第1のOSのｾﾏﾌｫｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ5030にｾｯﾄ
5560
第1のOSのｾﾏﾌｫｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ有効ﾌﾗｸﾞ5040をON
5565
指定されたｾﾏﾌｫの名前をｾﾏﾌｫの名前5020に格納
5570
ｴﾝﾄﾘ有効ﾌﾗｸﾞ5010をON
5575
第２のOS1300へのコンテキスト切替え
5580
セマフォ記述子をリターン
セマフォ記述子をリターン
5590

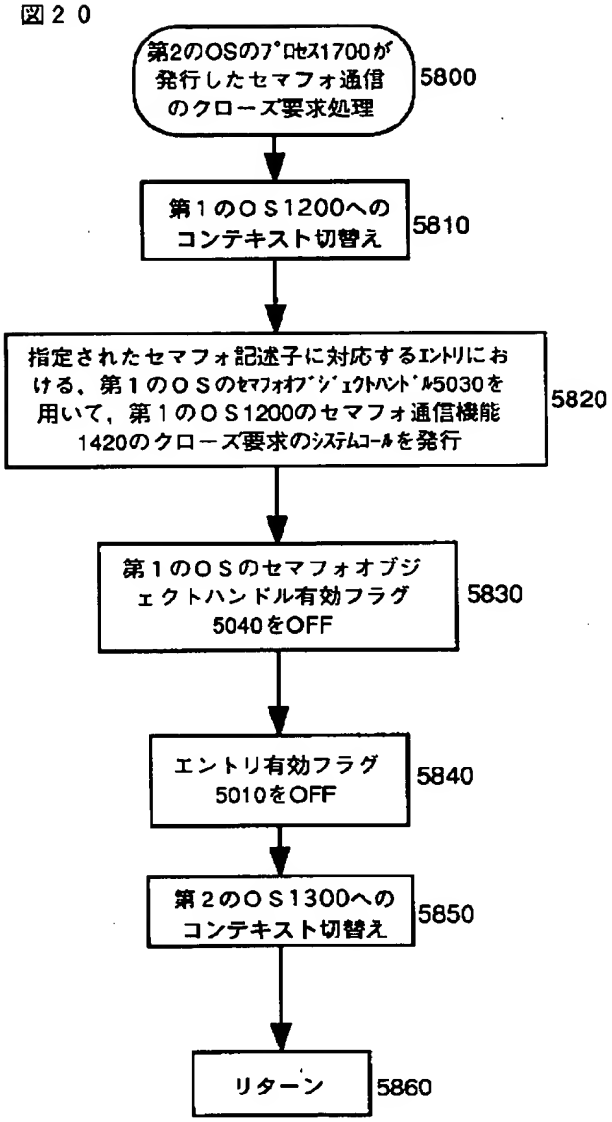
【図１９】

図１９



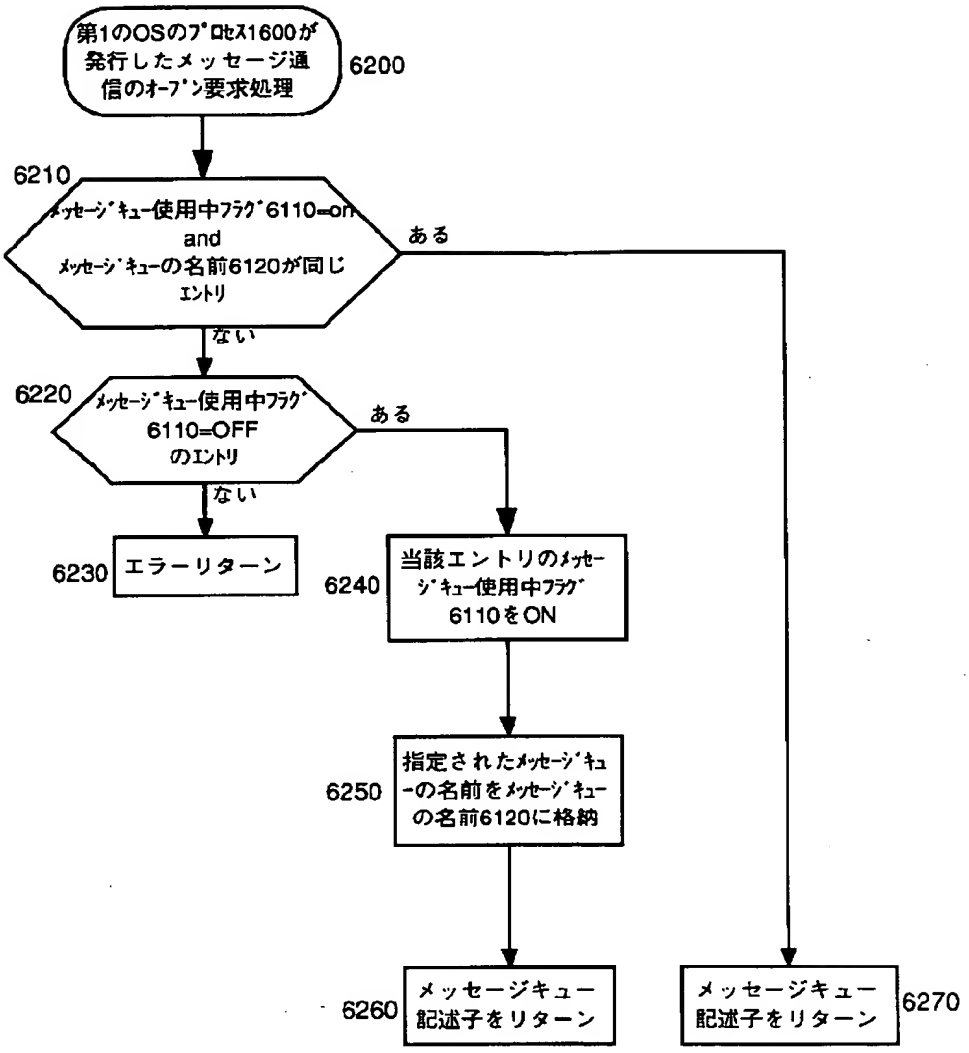
【図２０】

図２０



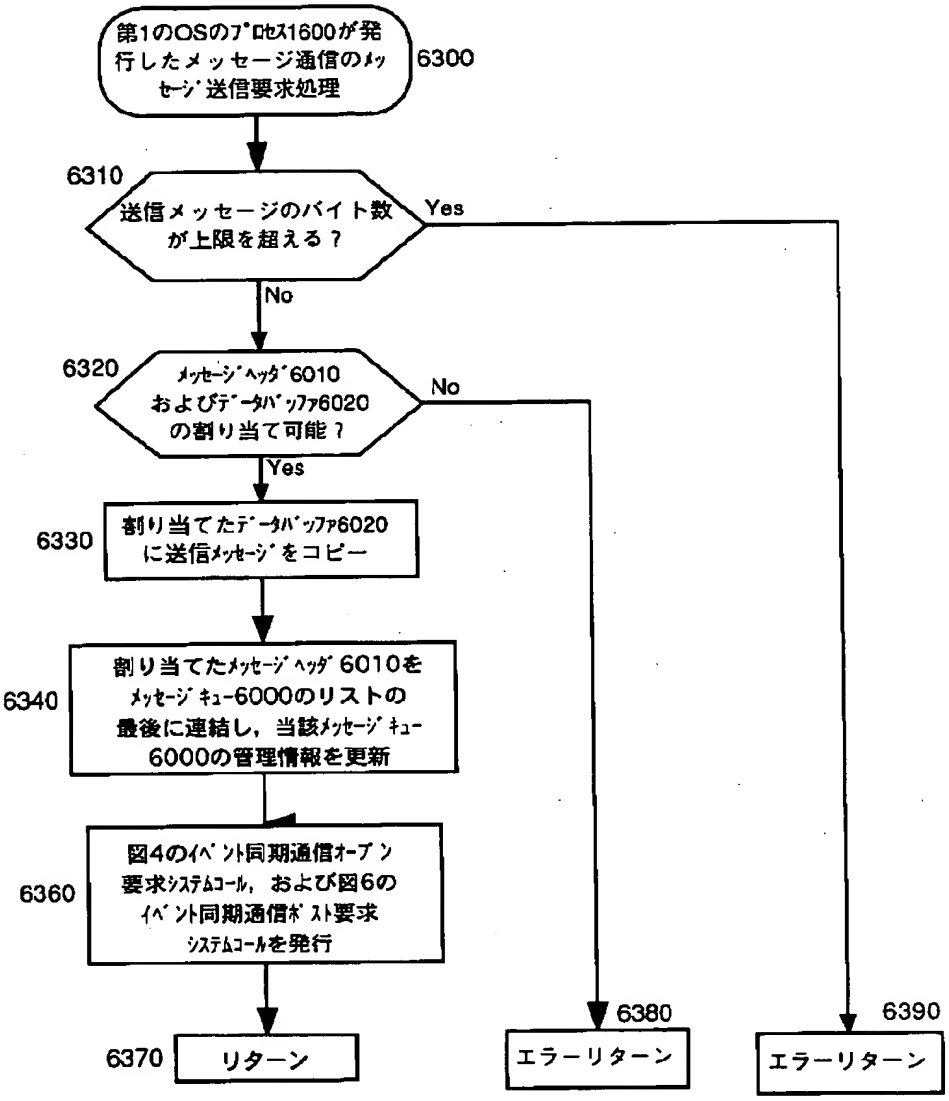
【図２３】

図２３


第1のOSのﾌﾟﾛｾｽ1600が発行したメッセージ通信のｵｰﾌﾟﾝ要求処理
6200
6210
ﾒｯｾｰｼﾞｷｭｰ使用中ﾌﾗｸﾞ6110=on and ﾒｯｾｰｼﾞｷｭｰの名前6120が同じエントリ
ある
ない
6220
ﾒｯｾｰｼﾞｷｭｰ使用中ﾌﾗｸﾞ6110=OFFのエントリ
ある
ない
6230
エラーリターン
6240
当該エントリのﾒｯｾｰｼﾞｷｭｰ使用中ﾌﾗｸﾞ6110をON
6250
指定されたﾒｯｾｰｼﾞｷｭｰの名前をﾒｯｾｰｼﾞｷｭｰの名前6120に格納
6260
メッセージキュー記述子をリターン
メッセージキュー記述子をリターン
6270

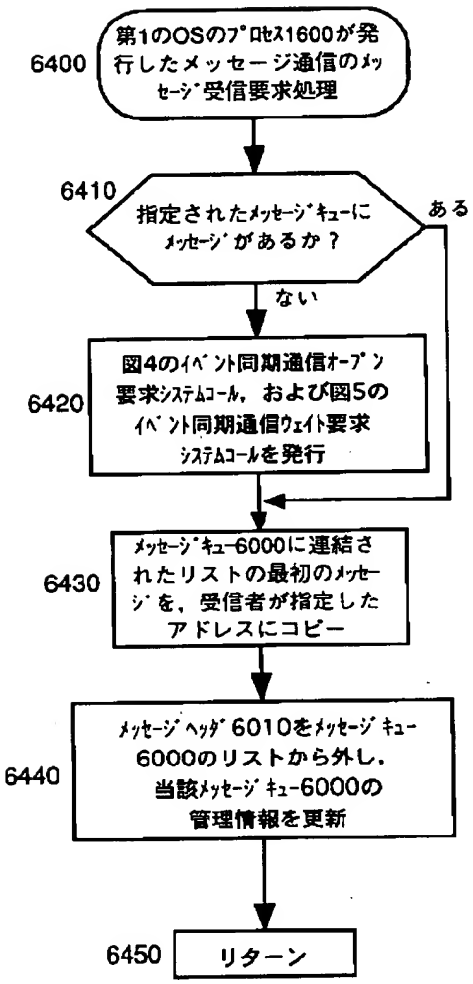
【図２４】

図２４


第1のOSのﾌﾟﾛｾｽ1600が発行したメッセージ通信のﾒｯｾｰｼﾞ送信要求処理
6300
6310
送信メッセージのバイト数が上限を超える？
Yes
No
6320
ﾒｯｾｰｼﾞﾍｯﾀﾞ6010およびﾃﾞｰﾀﾊﾞｯﾌｧ6020の割り当て可能？
No
Yes
6330
割り当てたﾃﾞｰﾀﾊﾞｯﾌｧ6020に送信ﾒｯｾｰｼﾞをコピー
6340
割り当てたﾒｯｾｰｼﾞﾍｯﾀﾞ6010をﾒｯｾｰｼﾞｷｭｰ6000のリストの最後に連結し，当該ﾒｯｾｰｼﾞｷｭｰ6000の管理情報を更新
6360
図4のｲﾍﾞﾝﾄ同期通信ｵｰﾌﾟﾝ要求ｼｽﾃﾑｺｰﾙ，および図6のｲﾍﾞﾝﾄ同期通信ﾎﾟｽﾄ要求ｼｽﾃﾑｺｰﾙを発行
6370
リターン
6380
エラーリターン
6390
エラーリターン

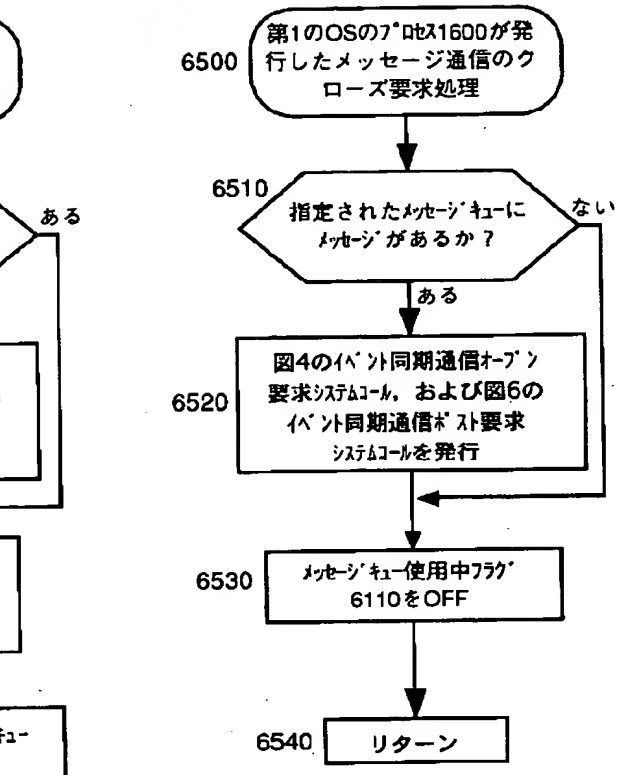
【図２５】

図２５



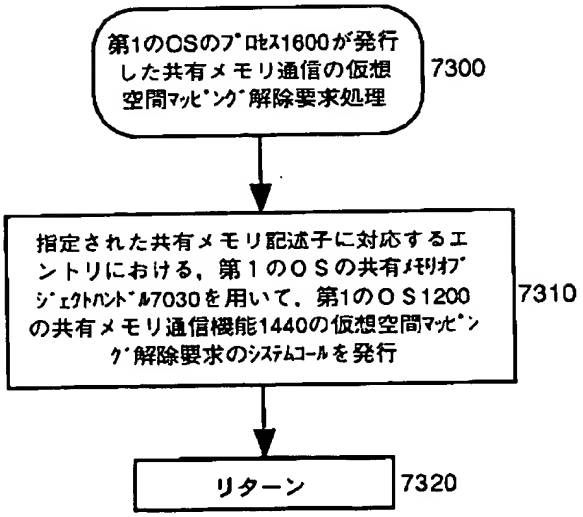
【図２６】

図２６



【図３４】

図３４



【図２７】

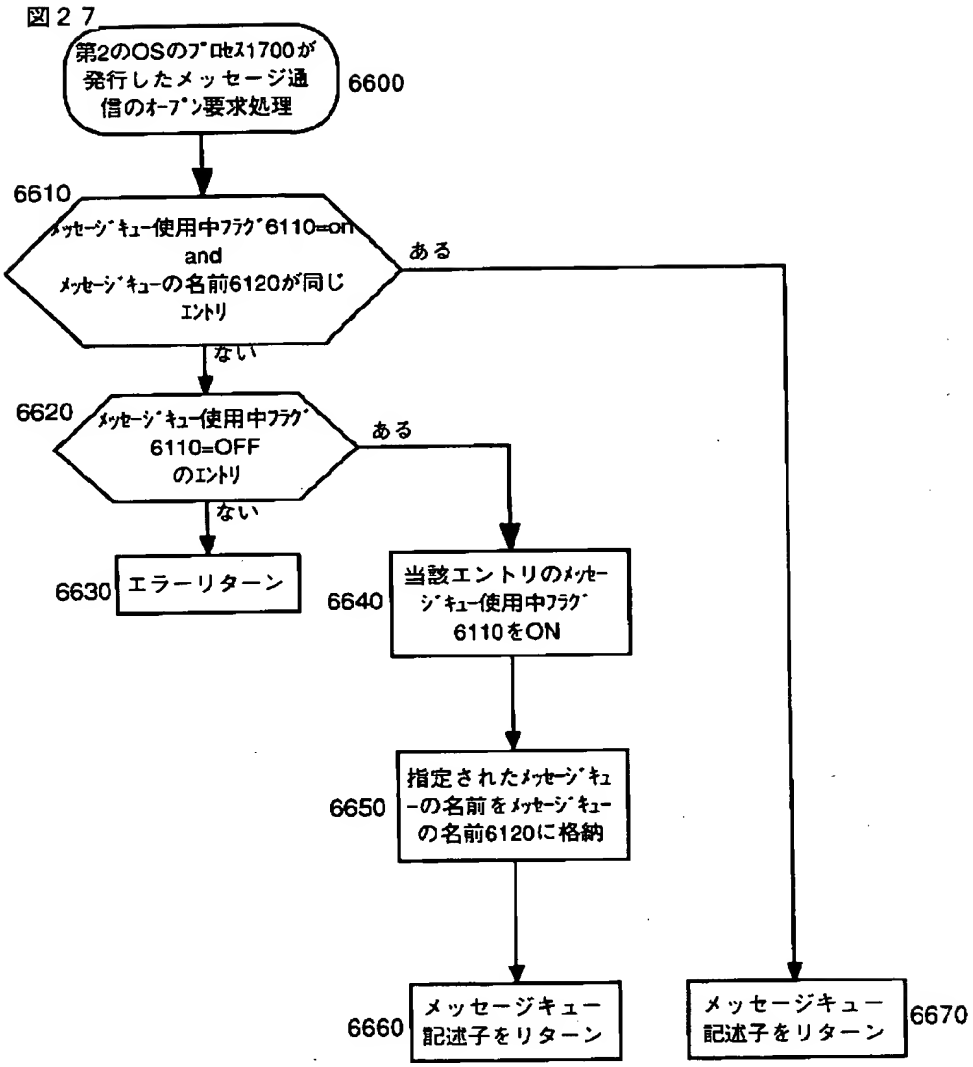
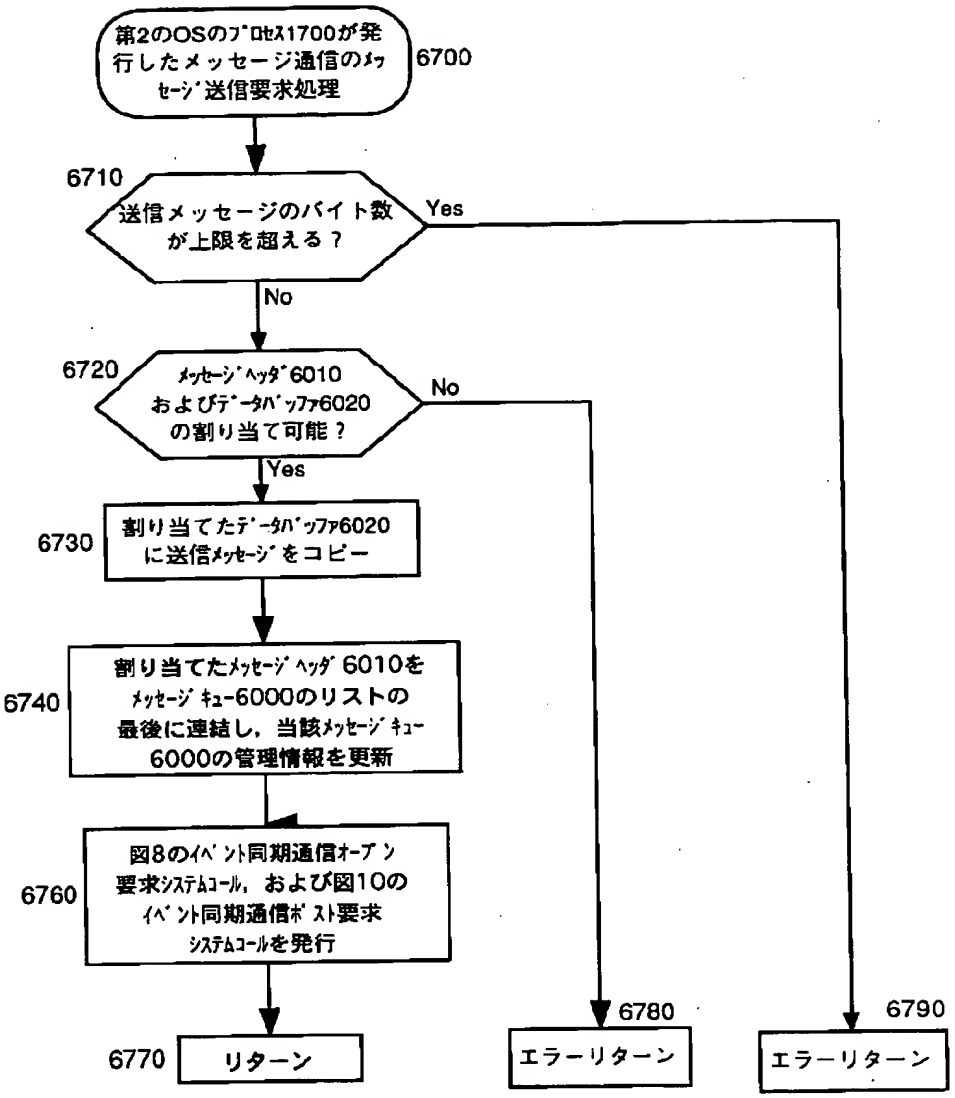


図２７
第2のOSのﾌﾟﾛｾｽ1700が発行したメッセージ通信のｵｰﾌﾟﾝ要求処理
6600
6610
ﾒｯｾｰｼﾞｷｭｰ使用中ﾌﾗｸﾞ6110=on and ﾒｯｾｰｼﾞｷｭｰの名前6120が同じエントリ
ある
ない
6620
ﾒｯｾｰｼﾞｷｭｰ使用中ﾌﾗｸﾞ6110=OFFのエントリ
ある
ない
6630
エラーリターン
6640
当該エントリのﾒｯｾｰｼﾞｷｭｰ使用中ﾌﾗｸﾞ6110をON
6650
指定されたﾒｯｾｰｼﾞｷｭｰの名前をﾒｯｾｰｼﾞｷｭｰの名前6120に格納
6660
メッセージキュー記述子をリターン
メッセージキュー記述子をリターン
6670

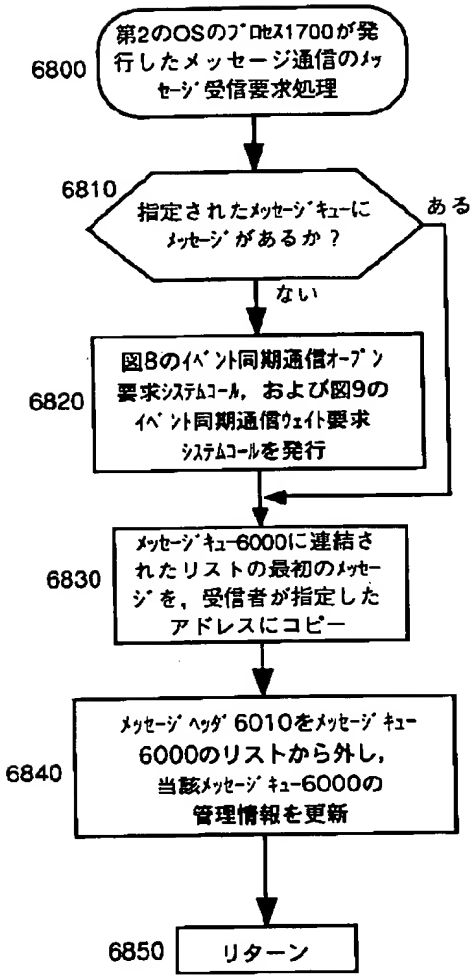
【図２８】

図２８


第2のOSのﾌﾟﾛｾｽ1700が発行したメッセージ通信のﾒｯｾｰｼﾞ送信要求処理
6700
6710
送信メッセージのバイト数が上限を超える？
Yes
No
6720
ﾒｯｾｰｼﾞﾍｯﾀﾞ6010およびﾃﾞｰﾀﾊﾞｯﾌｧ6020の割り当て可能？
No
Yes
6730
割り当てたﾃﾞｰﾀﾊﾞｯﾌｧ6020に送信ﾒｯｾｰｼﾞをコピー
6740
割り当てたﾒｯｾｰｼﾞﾍｯﾀﾞ6010をﾒｯｾｰｼﾞｷｭｰ6000のリストの最後に連結し，当該ﾒｯｾｰｼﾞｷｭｰ6000の管理情報を更新
6760
図8のｲﾍﾞﾝﾄ同期通信ｵｰﾌﾟﾝ要求ｼｽﾃﾑｺｰﾙ，および図10のｲﾍﾞﾝﾄ同期通信ﾎﾟｽﾄ要求ｼｽﾃﾑｺｰﾙを発行
6770
リターン
6780
エラーリターン
6790
エラーリターン

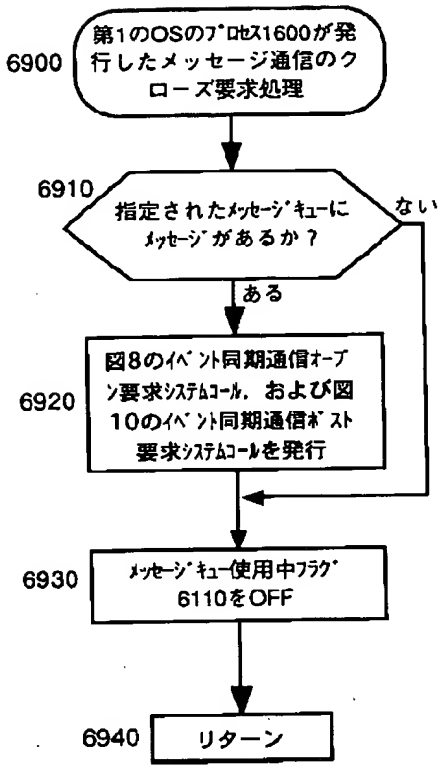
【図２９】

図２９



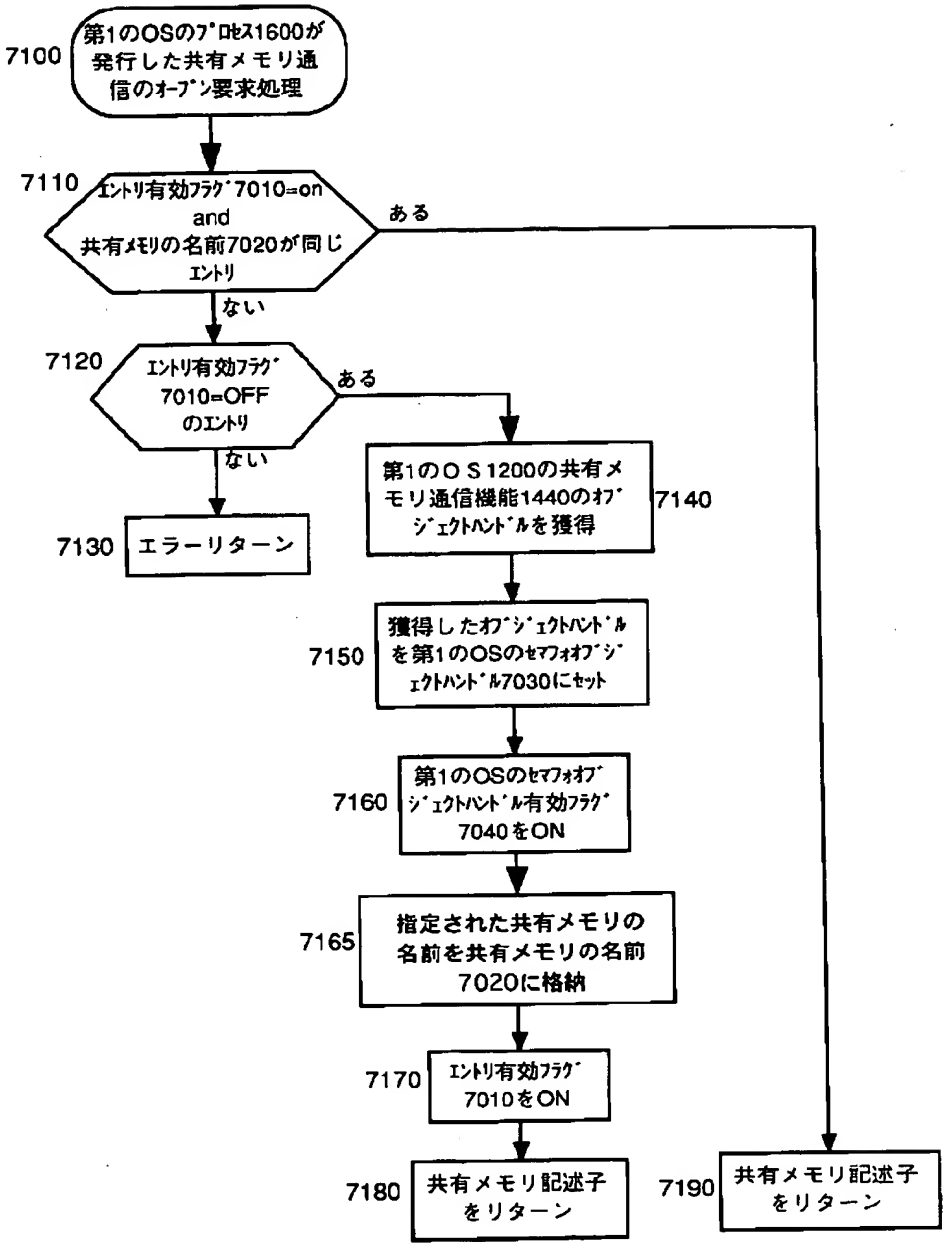
【図３０】

図３０



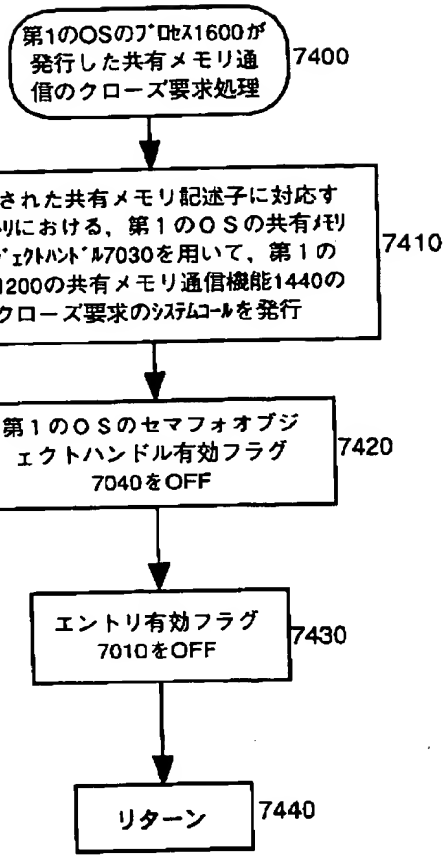
【図３２】

図３２


7100
第1のOSのﾌﾟﾛｾｽ1600が発行した共有メモリ通信のｵｰﾌﾟﾝ要求処理
7110
ｴﾝﾄﾘ有効ﾌﾗｸﾞ7010=on and 共有メモリの名前7020が同じｴﾝﾄﾘ
ある
ない
7120
ｴﾝﾄﾘ有効ﾌﾗｸﾞ7010=OFFのｴﾝﾄﾘ
ある
ない
7130
エラーリターン
第1のＯＳ1200の共有メモリ通信機能1440のｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを獲得
7140
7150
獲得したｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを第1のOSのｾﾏﾌｫｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ7030にｾｯﾄ
7160
第1のOSのｾﾏﾌｫｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ有効ﾌﾗｸﾞ7040をON
7165
指定された共有メモリの名前を共有メモリの名前7020に格納
7170
ｴﾝﾄﾘ有効ﾌﾗｸﾞ7010をON
7180
共有メモリ記述子をリターン
7190
共有メモリ記述子をリターン

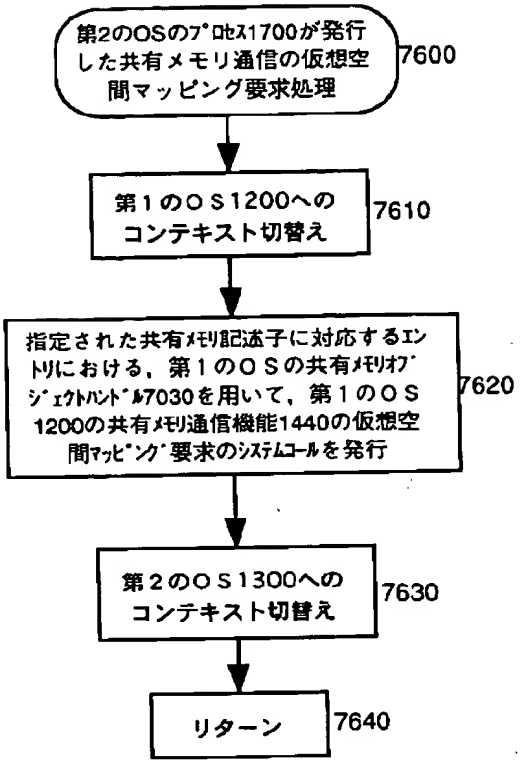
【図３５】

図３５



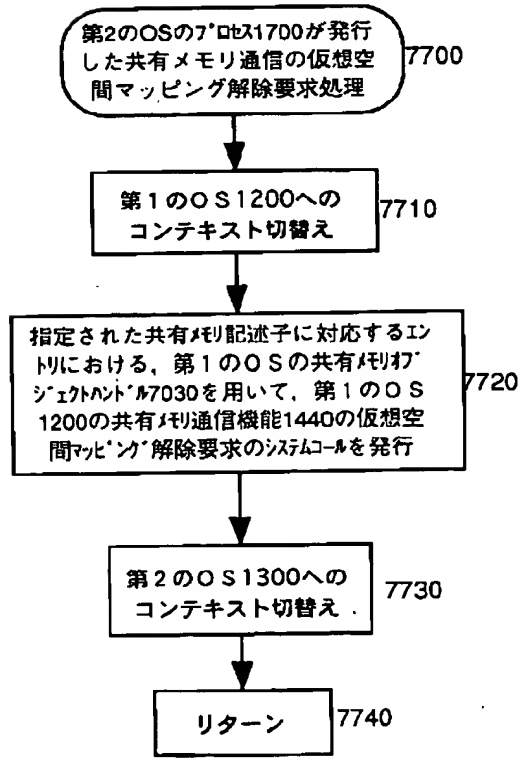
【図３７】

図３７



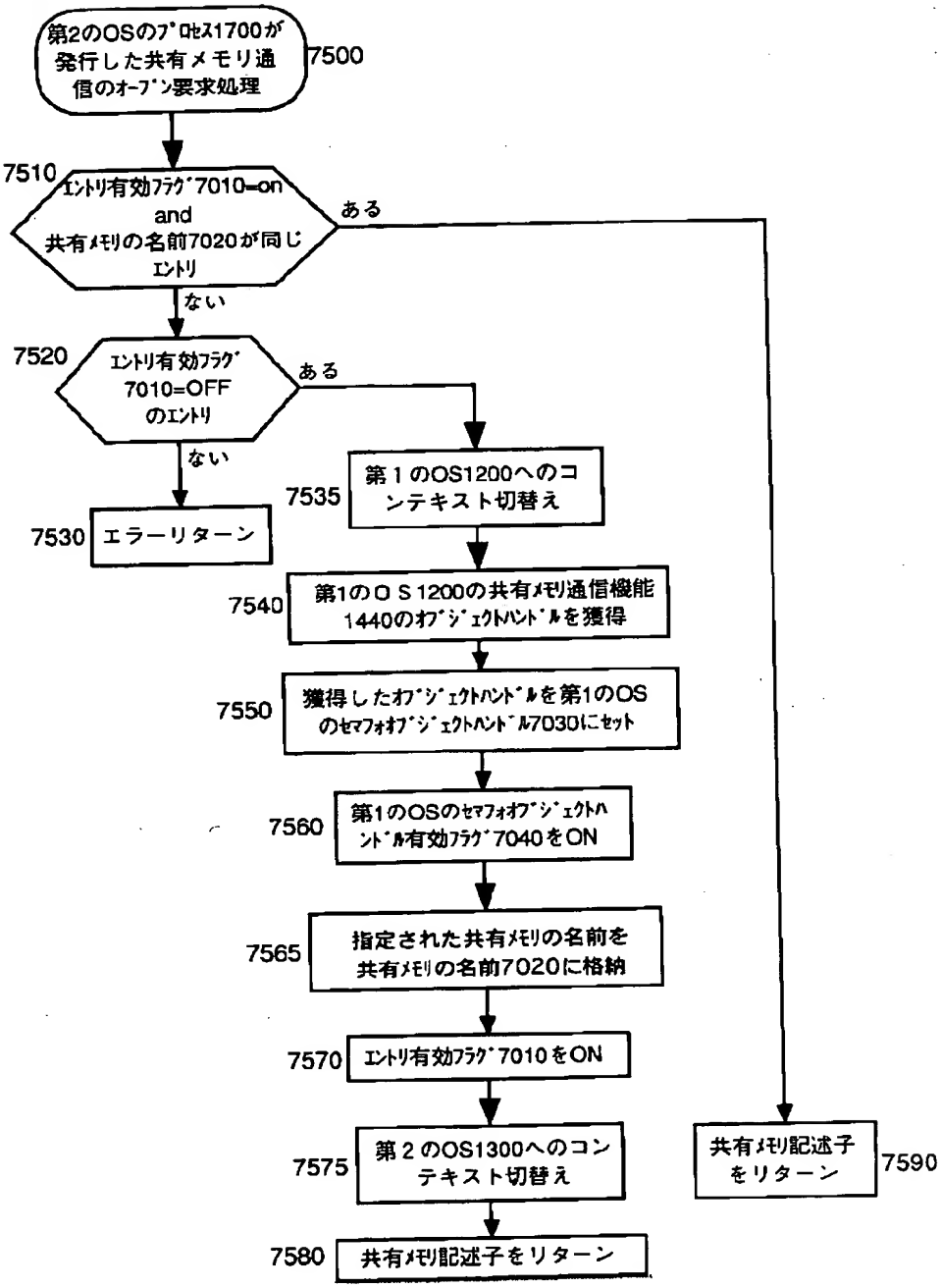
【図３８】

図３８



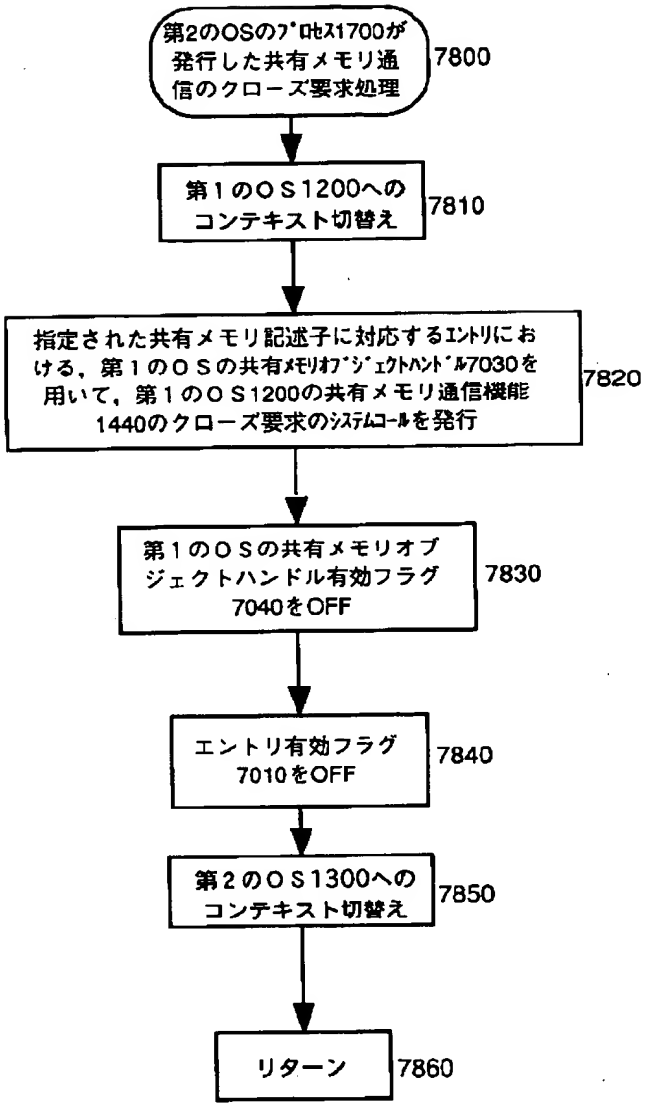
【図３６】

図３６


第2のOSのﾌﾟﾛｾｽ1700が発行した共有メモリ通信のｵｰﾌﾟﾝ要求処理
7500
7510
ｴﾝﾄﾘ有効ﾌﾗｸﾞ7010=on and 共有ﾒﾓﾘの名前7020が同じｴﾝﾄﾘ
ある
ない
7520
ｴﾝﾄﾘ有効ﾌﾗｸﾞ7010=OFFのｴﾝﾄﾘ
ある
ない
7530
エラーリターン
7535
第１のOS1200へのコンテキスト切替え
7540
第1のＯＳ1200の共有ﾒﾓﾘ通信機能1440のｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを獲得
7550
獲得したｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙを第1のOSのｾﾏﾌｫｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ7030にｾｯﾄ
7560
第1のOSのｾﾏﾌｫｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾙ有効ﾌﾗｸﾞ7040をON
7565
指定された共有ﾒﾓﾘの名前を共有ﾒﾓﾘの名前7020に格納
7570
ｴﾝﾄﾘ有効ﾌﾗｸﾞ7010をON
7575
第２のOS1300へのコンテキスト切替え
7580
共有ﾒﾓﾘ記述子をリターン
共有ﾒﾓﾘ記述子をリターン
7590

【図３９】

図３９



フロントページの続き

(72)発明者　大野　洋
茨城県日立市大みか町五丁目２番１号　株式会社日立製作所大みか工場内

(72)発明者　関口　知紀
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(72)発明者　柴田　隆
神奈川県横浜市戸塚区戸塚町5030番地　株式会社日立製作所ソフトウェア開発本部内